UNIVERZITA KARLOVA V PRAZE

Přírodovědecká fakulta

Katedra kartografie a geoinformatiky

Bc. Boris Navrátil

Metodika měření výkonnosti prostorových databází

Benchmark methodology of spatial databases

Diplomová práce

Praha 2016

Vedoucí diplomové práce: Ing. Miroslav Čábelka

Vysoká škola: Univerzita Karlova **Fakulta:** Přírodovědecká

Katedra: Aplikované geoinformatiky a kartografie **Školní rok:** 2016/2017

Zadání diplomové práce

Jméno a příjmení: **Boris Navrátil**

Datum a místo narození: **18. 3. 1992, Ostrava**

Studijní obor: **Kartografie a geoinformatika**

Garant studijního programu/oboru vám schválil přidělení této diplomové práce:

Název tématu: **Infrastruktura prostorových dat**

Zásady pro vypracování

Diplomová práce se bude zabývat problematikou měření výkonnosti prostorových databází. Hlavním cílem práce je navrhnout vlastní metodiku pro testování výkonu prostorových databází. Student se seznámí s problematikou databází a provede rešeršní průzkum druhů měření efektivity databázových systémů podporujících prostorová data.

Hlavní částí práce bude návrh vlastních testů pro ověření výkonu prostorových databází. Testy budou vhodně rozděleny dle typu do kategorií a následně bude navržen způsob jejich použití. Metodika testování výkonu bude navržena pro datový soubor volně stažitelných OpenStreetMap.

V praktické části bude provedeno měření výkonnosti podle navržené metodiky na třech vybraných různých databázových systémech. Výsledky testů budou mezi sebou vizuálně i metricky zhodnoceny.

Rozsah průvodní zprávy: cca 40 - 80 stran

Rozsah grafických prací: cca 5 - 10 stran

Seznam odborné literatury:

1) HUPPLER, Karl. 2009. The Art of Building Good Benchmark. First TPC Technology Conference, TPCTC 2009, Revised Selected Papers. Berlin.

2) WERSTEIN, Paul. 1998. A performance Benchmark for Spatiotemporal Databases. Dunedin. 9 s, University od Otago.

3) TRODD, N. 2012. Spatial Data Models and Spatial Data Structures. 33 s, Coventry University.

4) JEDLINSKÝ, Jan. 2006. Způsoby uložení prostorových dat v databázi pro účely pozemkového datového modelu. Plzeň. 56 s. Diplomová práce. Západočeský univerzita.

5) KOLÁŘ, Dušan. 2006. Pokročile databázové systémy (Prostorové databáze). FIT VUT v Brně.

6) REID, G. Elizabeth. 2009. Design and Evaluation of a Benchmark for Main Memory Transaction Processing Systems. Massachusetts. 63 s., Diplomová práce. Massachusetts institute of technology.

Vedoucí diplomové práce: **Ing. Miroslav Čábelka**

Konzultant diplomové práce:

Datum zadání diplomové práce: 19. 1. 2015

Termín odevzdání diplomové práce: květen 2017

V Praze dne 17. 2. 2017

…….. ……

Vedoucí diplomové práce Garant oboru

Prohlášení

Prohlašuji, že jsem předkládanou diplomovou práci vypracoval samostatně, všechny použité prameny a literatura byly řádně citovány a práce nebyla využita k získání jiného nebo stejného titulu.

V Praze, dne 15. dubna 2017

.....

Podpis

Poděkování

Na tomto místě bych rád poděkoval vedoucímu své diplomové práce, Ing. Miroslavu Čábelkovi za odborné vedení, poskytnutí literatury i čas, který se mnou strávil na konzultacích i při opravách textů. Dále děkuji Ing. Marku Kocanovi za konzultační činnost v počátcích práce. V neposlední řadě děkuji také svým rodičům, kteří mi studium na vysoké škole umožnili.

Abstrakt

Diplomová práce se zabývá porovnáváním výkonnosti databázových systémů, které umožňují práci s prostorovými daty. Měření výkonnosti probíhá nad daty OpenStreetMap. Velký důraz je kladen na tvorbu metodiky měření výkonnosti databázových systémů. Metodika je navržena tak, aby byla aplikovatelná na všechny druhy databázových technologií a aby její životnost byla aktuální pro dalších několik let. Metodika je odzkoušená na 3 databázových systémech (PostgreSQL, MySQL a SQLite) a výsledky jsou metricky, graficky a verbálně vyhodnoceny.

Klíčová slova: benchmark, testování, prostorová databáze, MySQL, PostgreSQL, PostGIS, SQLite, SpatiaLite, OpenStreetMap, OGC.

Abstract

This master‘s thesis deals with comparing the performance of database systems that allow work with spatial data. Performance measurement takes place over data OpenStreetMap. Considerable emphasis is put on the creation of the methodology for database systems efficiency measurement. The methodology is designed for all kinds of database technologiesto ensure its applicability for the next years. The method has been tested on three database systems (PostgreSQL, MySQL and SQLite) and the results are metrically, verbally and graphically evaluated.

Keywords: benchmark, testing, spatial database, MySQL, PostgreSQL, PostGIS, SQLite, SpatiaLite, OpenStreetMap, OGC.

OBSAH

SEZNAM SYMBOLŮ A ZKRATEK

ACID	Atomicity, Consistency, Isolation, Durability
CAD	Computer-Aided Design
CAM	Computer-Aided Manufacturing
CPU	Central Processing Unit
DCL	Data Control Language
DDL	Data Definition Language
DML	Data Manipulation Language
GDAL	Geospatial Data Abstraction Library
GIS	Geographic Information Systems
GPL	General Public License
GWPG	Graphic and Workstation Performance Group
HPG	High Performance Group
IS	Information System
MBR	Minimum Bounding Rectangle
NoSQL	Not Only SQL
OGC	Open Geospatial Consortium
OOP	Object-oriented programming
OOSŘBD	Objektově-Orientovaný Systém Řízení Báze Dat
OSG	Open System Group
OSM	Open Street Map
SFS	Simple Feature Standard
SQL	Structured Query Language
SPEC	Standard Performance Evaluation Corporation
SŘBM	Systém Řízení Báze Dat
TPC	Transaction Processing Performance Council
WKB	Well-Know Binary
WKT	Well-Know Text

SEZNAM OBRÁZKŮ A GRAFŮ

SEZNAM TABULEK

SEZNAM PŘÍLOH

ÚVOD

Jednou z nejdůležitějších úloh počítačů je ukládání počítačově zpracovávaných dat. Existuje mnoho způsobů, jak tato data ukládat. Nejpraktičtější způsob je tato data ukládat pomocí databází. Ukládání dat do databází má mnoho předností, které jsou v práci popsány.

V současné době je 95 % všech aplikací databázových a toto procento stále narůstá. Úměrně s nárůstem zpracování dat v databázích, narůstá i potřeba ukládat prostorová data v databázích. Prostorová data jsou velice specifická a k práci s těmito daty jsou zapotřebí specifické databázové systémy. V teoretické části práce jsou čtenářům přiblíženy různé typy databází, prostorová data a organizace, která má za cíl standardizovat práci s prostorovými daty.

Drtivá většina aplikací je databázových, proto je potřeba nalézt vhodný typ databáze pro konkrétní úlohu. Jedním ze způsobů nalezení vhodných databázových systémů slouží testy, které tyto databáze porovnávají. V teoretické části práce je zahrnutý rešeršní průzkum měření efektivity databázových systémů a požadavky pro tvorbu dobré metodiky měření výkonnosti databázových systému.

I přes obrovskou komunitu uživatelů zabývající se databázemi existuje velice málo metod, jak měřit výkon jednotlivých databází. Mezi nejznámějšími testy dostupnými na trhu jsou testy organizací TPC a SPEC, které se zabývají měřením výkonnosti databází transakčního zpracování. V oblasti prostorových databází je tato problematika ještě chudší a s narůstajícím počtem prostorových dat je tento nedostatek nezanedbatelný. Velká pozornost v diplomové práci je věnovaná tvorbě metodiky na porovnávání výkonu prostorových databází. Metodika porovnávání je navržena pro práci s datovým souborem volně stažitelných OpenStreetMap.

Cílem diplomové práce je navrhnout dostatečně přesnou a široce využitelnou metodiku měření výkonnosti databázových systémů, která testuje prostorové funkce nad datovou sadou OpenStreetMap. Cílem metodiky je navržení testů, které budou platné po dobu několika dalších let a také dané testy musí být aplikovatelné na nové databázové technologie.

Ke splnění těchto cílů je potřeba dostatečně porozumět aktuálním databázovým řešením, které jsou momentálně na trhu.

Dalším cílem je navrženou metodiku otestovat na 3 různých databázových systémech a výsledky mezi sebou vizuálně i metricky porovnat.

Praktická část práce demonstruje vytvořenou metodiku měření výkonností na 3 různých databázových systémech na linuxové architektuře virtuálního serveru. Výsledky jsou metricky zhodnocené formou tabulek a jednotlivé databáze jsou mezi sebou porovnány pomocí grafů.

1. DATABÁZE

Databázi je možné popsat jako strukturovaný soubor záznamů (dat), které jsou mezi sebou navzájem propojeny pomocí klíčů. Strukturovanost dat je dosažena pomocí tzv. datového modelu. Nejznámější používané datové modely jsou: (WON, 1995)

- Hierarchický
- Síťový
- Relační
- Objektový
- Objektově-relační
- NoSQL

1.1. Hierarchický datový model

Data v tomto modelu jsou hierarchicky strukturovaná a lze si tento model představit jako obrácený strom. Vztahy v databázi jsou reprezentovány termíny rodič a potomek. Tento datový model je specifický tím, že rodič může mít více potomků, ale potomek může mít právě jednoho rodiče. K záznamům se přistupuje od kořene modelu (tabulka, která neobsahuje rodiče, existuje právě jedna) až po listové prvky (záznamy, které už neobsahují potomky). (MERTZ, 2001)

Výhodami tohoto modelu je snadné přidávání nových položek do databáze a flexibilita ve volbě hranic. Nevýhody tento model má v nezajištění jednoduché metody pro tvorbu vztahů M : N a složité operace pro vkládání nebo rušení záznamů.

1.2. Síťový datový model

Data v tomto modelu jsou vyjádřena pomocí uzlů a vztahů mezi těmito uzly. Uzel reprezentuje soubor záznamů. Tato metoda je vylepšením vztahu rodič/potomek (hierarchického modelu). Tento model je snadno pochopitelný, jeden uzel je definován jako vlastník a druhý jako prvek. Množinová struktura podporuje vztah 1 : N (jeden vlastník může obsahovat více prvků, ale prvek je ve vztahu pouze k jednomu vlastníkovi). (VALENTA, 2011)

Výhodami síťového modelu je rychlý přístup k datům, omezení vzniku některých

duplicitních dat a snazší metoda pro tvorbu vazeb M : N. Nevýhodou je, že uživatel pro práci s množinovými strukturami musí znát strukturu databáze.

1.3. Relační datový model

Tento model byl poprvé popsán Dr. Coddem v roce 1970 a v současné době se jedná o nejčastěji využívaný SŘBD. Data v tomto modelu jsou vyjádřena pomocí relací. Relaci si lze představit jako tabulku (dvourozměrná struktura tvořená záhlavím a tělem). Každý sloupec tabulky musí mít název a datový typ pro ukládání určitých dat. Každý řádek tabulky představuje jeden záznam (výskyt). Každý záznam (řádek) musí mít jednoznačný identifikátor (primární klíč) (VALENTA, 2011), (SILBERSCHATZ a kol., 2006). Jedná se o přesně definovaný matematický model postavený na rozšířené výrokové logice a teorii množin. Veškeré relační databáze jsou de facto implementací modelu, který opisuje Coddův matematický model a velice obecně řečeno, jazyk SQL nahrazuje matematické výrazy. (SHAMKANT, 2003)

Základním prvkem relačních databází jsou databázové tabulky. Tyto tabulky se skládají ze sloupců (atributů) a řádků (záznamů). Atributy ukládají datum v konkrétním datovém typu – doméně. Řádek (tuple) slouží k uložení dat. Konkrétní tabulka pak realizuje podmnožinu kartézského součinu možných dat všech sloupců. (CODD, 1970), (SHAMKANT, 2003)

Výhodou je flexibilita mezi různými záznamy, naopak nevýhodou relačního modelu je nepřirozený popis některých objektů reálného světa. Relační model také neumožňuje ternární vazbu mezi entitami.

1.4. Objektový datový model

Je datový model, ve kterém jsou data reprezentována ve formě objektů, stejně jako jsou použita v objektově-orientovaném programování (PRABHU, 1998). V roce 1993 definovalo konsorcium vůdčích výrobců OOSŘBD standard ODMG-93. Hlavní komponenty tohoto standardu jsou: (CATTELL, 1994)

- Objektový model
- Objektový jazyk pro definici dat
- Objektový jazyk pro dotazy na data
- Vazba na jazyk C++

- Vazba na jazyk Smalltalk

Objektově orientovaný systém řízení báze dat (OOSŘBD) kombinuje databázové schopnosti se schopnostmi objektově orientovaného programování. OOSŘBD umožňuje programátorům ukládání objektů, replikování nebo upravování stávajících objektů, z nichž vznikají nové objekty. (PRABHU, 1998)

Mezi výhody objektového modelu patří lepší podpora datových struktur z OOP, není potřeba transformace datových struktur k ukládání do databází a oproti relačním databázím předpokládáme efektivnější zpracování dotazu (OO datový model navazuje na síťový).

1.5. Objektově – relační model

Jedná se o systém řízení báze dat podobný relační databázi, ale s objektově orientovaným datovým modelem (objekty, třídy, dědičnost je podporována v databázovém schématu a v dotazovacím jazyce). Objektově – relační databázové systémy podporují rozšíření datového modelu vlastními datovými typy a metodami. (POKORNÝ a VALENTA, 2005)

Objektově – relační databáze poskytují střední cestu mezi relačními databázemi a objektově orientovanými databázemi. Výhodou těchto databázových systémů je kombinace relačních i objektových vlastností databází dohromady. Možnost přidávání nových datových typů a funkcí.

1.6. NoSQL

Jedná se o širokou skupinu nerelačních databází. Tyto databáze většinou nepoužívají pro dotazování jazyk SQL na rozdíl od relačních databází. Termín NoSQL bývá vysvětlován databázovou komunitou jako „not only SQL“. (SHASHANK, 2011)

Do této skupiny databází patří mnoho datových uložišť. Od grafových, dokumentových, objektových až po XML databáze. Nejčastěji se využívá ukládání dat jako kombinace dvojic klíč/hodnota. V tomto modelu je klíč shodný s názvem sloupce v relační databázi. Výhody tohoto modelu jsou v rychlosti, škálovatelnosti a nepotřebě hodnot NULL. Jako nevýhoda je považován příliš jednoduchý datový model. (SHASHANK, 2011)

1.7. Jazyk SQL

Jedná se o standardizovaný strukturovaný programovací jazyk SQL (Structure Query Language) používaný primárně pro práci s daty v relačních databázích.

1.7.1. Historie

První příkazy pro ovládání relačních databází vznikly v 70. letech 20. století firmou IBM. Vytvořený jazyk nesl pojmenování SEQUEL (Structured English Query Language). Hlavním cílem bylo vymyslet jazyk, který se bude co nejblíže podobat přirozenému jazyku. K vývoji jazyka se postupně přidávaly další firmy (Oracle database, SyBase) a SEQUEL byl přejmenovaný na SQL. Se stále větším rozšířením relačních databází bylo nutné jazyk pro komunikaci s těmito databázemi standardizovat. Proto v roce 1986 americký institut ANSI založil standard s označením SQL-86. S potřebami nových funkcí byl v roce 1992 přijat nový standard (SQL-92), který nese nové prvky, týkající se převážně integrity databáze. Zatím posledním standardem je SQL-99, který reaguje na moderní trend databází s objektovými prvky. (RYDVAL, 2005)

1.7.2. Popis

Základní datové typy jsou podle (OLSZOWSKI a FARANA, 1997) rozdělené do několika kategorií:

- Řetězcové: CHARACTER(n), CHARACTER VARYING(n), VARYING(n)
- Numerické
  - Přesné - NUMERIC(p, q), DECIMAL(p, q),
  - Přibližné – INTEGER, SMALLINT, FLOAT(p), REAL, DOUBLE PRECISION
- Datum a čas: DATE, TIME, TIMESTAMP
- Intervalové: INTERVAL
- Booleovský: BIT, BOOLEAN

Základní příkazy jazyka SQL se dělí do několika skupin. První skupinou jsou příkazy pro manipulaci s daty, druhou skupinou jsou příkazy pro definici dat a třetí skupinou jsou příkazy pro kontrolu přístupu k datům. (OLSZOWSKI a FARANA, 1997)

Základní příkazy pro manipulaci s daty – DML (Data Manipulation Language).

Příkazy, které umožňují získávání, upravování, vkládání nebo mazání dat z databáze.

- SELECT – nejzákladnější příkaz, umožňuje výběr dat z databáze.
- INSERT – tento příkaz se používá pro vkládání nových záznamů do databáze.
- UPDATE – příkaz používaný pro editaci stávajícího záznamu.
- DELETE – příkaz pro mazání záznamu z databáze.

Základní příkazy pro definici dat – DDL (Data Definition Language).

Příkazy, které umožňují vytváření, editace a mazání objektů na databázích. Objektem se nemyslí záznam na tabulce, ale samotná tabulka, index nebo například pohled.

- CREATE – příkaz k vytvoření nového objektu, např. tabulky.
- ALTER – příkaz k editaci stávajícího objektu.
- DROP – příkaz k odstranění stávajícího objektu.

Příkazy pro kontrolu přístupu k datům – DCL (Data Control Language).

Jedná se o příkazy k řízení transakcí, programové příkazy (např. nastavování práv), agregované funkce a predikáty.

1.8. Prostorová data

V Terminologickém slovníku zeměměřictví a katastru nemovitostí (PAVELKA a kol., 2005) jsou prostorová data definovaná jako „*Data o poloze, tvaru a vztazích mezi jevy reálného světa, vyjádřená zpravidla ve formě souřadnic a topologie*“.

V geoinformatice u prostorových dat můžeme rozlišovat část polohovou (většinou souřadnice) a popisnou (vlastnosti prvku). Pro prezentování těchto dat existuje několik modelů prostorových dat. V praxi se k prezentování prostorových dat do digitální podoby setkáváme s 2 modely: rastrový a vektorový. (BŘEHOVSKÝ a JEDLIČKA, 2016)

1.8.1. Vektorový model

Vektorem je orientovaná úsečka, která je definovaná počátečním a koncovým bodem. Základními stavebními prvky vektorového modelu jsou: (BŘEHOVSKÝ a JEDLIČKA,

2016)

- Bod (Point) – bezrozměrný objekt, jednotlivý pár souřadnic X, Y, Z.
- Linie (Line) – 1D prvek, sled orientovaných křivek/hran, souřadnice jednotlivých vrcholů mezi dvěma uzly.
- Polygon – 2D objekt, hranice tvoří linie.

S využitím vektorové grafiky lze reprezentovat mnoho vektorových modelů, tím nejjednodušším je špagetový model, který má tu vlastnost, že každý objekt na mapě je reprezentován jedním záznamem (řetězcem X a Y souřadnic). Dalším modelem je topologický, který má vlastnost, že každá linie začíná, končí nebo se protíná s jinou linií v uzlu. Linie je reprezentována souborem souřadnic X a Y a v topologickém modelu jsou uchovávány informace, který polygon leží vpravo a který vlevo vzhledem k linii. Posledním modelem je hierarchický, který samostatně ukládá informace o bodech, liniích a plochách v hierarchické struktuře pro jednodušší vyhledávání v datech. Tento model je pro vyhledávání a manipulaci s daty nejvhodnější. (TRODD, 2012)

1.8.2. Rastrový model

Je složen z pravidelné nebo nepravidelné sítě či mozaiky. Jednotkou této sítě je buňka (pixel, cell). Na rozdíl od vektorové reprezentace dat se rastrová zaměřují na zemský povrch nebo lokalitu jako celek. Tímto modelem se spíše reprezentují jevy, které jsou definované na celém území. (TRODD, 2012)

Výhoda rastrového modelu je lepší reprezentace dané lokality, např. terénní reliéf. Další výhodou je jednodušší tvorba některých analýz a výpočtů. Nevýhodou rastrového modelu je výpočetní i algoritmická náročnost. (TRODD, 2012)

Reprezentace pomocí vektorového modelu je vhodnější pro prvky bodového, liniového či plošného (nespojitého) charakteru. Pomocí rastrového modelu je vhodné reprezentovat plošné spojité jevy, např. fyzikální veličiny. (BŘEHOVSKÝ a JEDLIČKA, 2016)

1.9. Prostorové databáze

Prostorovou databází můžeme označit databázový systém s podporou ukládání a zpracování prostorových dat. Prostorovými daty označujeme taková data, která se vážou

k určitému prostoru. Příkladem můžou být geometrické entity, které modelují řeky, silnice, lesy anebo domy. Příkladem prostoru může být souřadnicový systém, který má většinou dvě nebo tři dimenze.

Deskriptor (realm), jedná se o dynamickou, uživatelem definovanou základní strukturu datových typů. Příkladem u prostorových databází je deskriptor, definován jako planární graf nad diskrétní mřížkou, který má tyto vlastnosti:

- Každý samostatný bod je bodem diskrétní mřížky.
- Každý koncový bod linie nebo složitějšího objektu je bodem diskrétní mřížky.
- Žádný vnitřní bod linie nebo polygonu není zaznamenán v mřížce.
- Žádné dva složitější objekty nemají žádné společné body kromě koncových.

Prostorové datové typy a operace definuje prostorová algebra. Definice prostorových datových typů by měla splňovat několik kritérií. Prvně musí být u datových typů a funkcí zohledněna aritmetika s konečnou přesností. Dalším kritériem definice prostorových typů je podpora konzistentnosti popisu topologie souvisejících objektů. Posledním kritériem je nezávislost na konkrétním SŘBD, ale zároveň úzká spolupráce s tímto SŘBD.

V zásadě rozlišujeme 3 kategorie prostorových operací, které je možné na prostorové datové typy aplikovat.

- Predikáty – operace, které mají nějakou prostorovou vazbu mezi 2 prvky a výsledkem je pravdivostní hodnota. Příkladem jsou funkce *intersects* nebo *overlaps*.
- Geometrické relace – operace, které mají nějakou prostorovou vazbu mezi 2 prvky a výsledkem je jedna či více hodnot prostorového typu. Příkladem jsou funkce *union* nebo *intersection*.
- Výpočetní relace – operace, které vypočítají z jednoho nebo více objektů nějakou hodnotu nebo vytvoří nový prvek. Příkladem jsou funkce *buffer* nebo *area*.

Kromě těchto operací prostorové databáze musí podporovat i operace prostorové selekce (výběr založený na prostorovém predikátu) a prostorové spojení (spojení založené na predikátu testující prostorové atributy).

1.9.1. Databázové indexy

Databázové indexy slouží k rychlejší a efektivnější práci s daty. Vytvořením indexu databázový server ukládá informaci o rozmístění hodnot indexovaných sloupců v tabulce a při následném dotazování nemusí databázový server prohledávat tabulku sekvenčně, ale díky indexům, které jsou ukládané v paměťovém prostoru, se přistupuje přímo k relevantním řádkům tabulky. Indexy by se měly využívat u všech sloupců tabulek, díky kterým se vyhledává, třídí anebo spojuje obsah s další tabulkou. Základní představa indexů je reprezentovaná kombinací vyhledávacího klíče a ukazatele do tabulky. Speciálním typem indexu je primární klíč, který slouží v tabulce k jedinečnému identifikování záznamu. Téměř všechny databázové systémy nyní umožňují přidání indexu k dané tabulce. Existuje mnoho druhů indexů, nejčastěji se implementují pomocí B-stromu, které dosahují časové složitosti $O(\log N)$, kde N je počet prvků. (JEDLINSKY, 2006), (NEUBAUER, 1999), (VALENTA, 2016)

Indexy nemají jenom samá pozitiva. Jejich nevýhodou je potřeba určité režie ze strany databází při vkládání nových záznamů do tabulky a následný přepočet indexů. Proto se indexy nedoporučují ve specifických příkladech, například u logovacích tabulek. Dále indexy zabírají určité místo v paměti, a proto je potřeba indexy vytvářet s rozumem podle určitých pravidel. (KANDASAMY, 2015)

Dalšími speciálními typy indexů jsou prostorové indexy, které se používají k indexaci prostorové složky dané tabulky. Dělí se podle způsobu práce s výchozím prostorem na indexy, které snižují nebo zvyšují dimenze a indexy, které dělí oblasti na překrývající se nebo nepřekrývající se podoblasti. Typickým zástupcem prvního typu je linearizace. Za typické zástupce druhého typu se považují různé stromové struktury (např. K-d-stromy nebo R-stromy). (VALENTA, 2016), (POKORNÝ, 2000)

1.10. Open Geospatial Consortium

Jedná se o mezinárodní organizaci založenou za účelem standardizace geoprostorových dat, služeb, geoinformačních systémů a práce s daty. V OGC je zahrnováno více než 400 různých komerčních, dobrovolnických, vládních i vědeckých organizací. Cílem organizace je vybudovat specifikace pro běžné potřeby geoinformačních technologií (OGC about, 2016). Nyní existuje 28 různých specifikací, z nichž nejzákladnější se jmenuje Obecná

specifikace (Abstract Specification), která popisuje základní datový model pro geografické prvky. Základní myšlenkou standardu je, že prvek je považován za jednoduchý, nekříží-li se se sebou samým. Další OGC specifikace popisují mapové služby, katalogové služby nebo formáty pro uchovávání dat. V této práci je nejdůležitější Simple Features – SQL specifikace. (OGC standards, 2016)

Standard Simple Features specifikuje ukládání prostorových dat do digitální podoby. Dále tento standard popisuje různé prostorové funkce nebo operátory, kterými lze zjišťovat informace o daných prvcích, nebo kterými lze vytvářet nové prvky. V tomto dokumentu je také definovaný textový značkovací jazyk (Well-know text), který slouží k popisu vektorové geometrie, referenčních systémů nebo parametrů pro transformaci mezi jednotlivými souřadnicovými systémy. Pro ukládání informací v databázových systémech se používá binární forma (Well-know binary) (OGC standards, 2016). Metodika testovacího scénáře použita v diplomové práci vychází z tohoto standardu.

Většina prostorových rozšíření databázových systémů podporuje standardizaci Simple Features, a tak je vhodné funkce z tohoto standardu použít pro testovací scénář prostorových databází.

1.11. DE-9IM

Rozšířený rozměrový 9-ti průsečíkový model neboli Clementiniho matice, prezentuje vztahy mezi 2 prostorovými objekty. Každý objekt má vnitřní část, vnější část a hranici, společně zkombinují 9 případů, které mohou nastat.

Z tohoto modelu vychází prostorové vztahy, které jsou založené na binárních prostorových predikátech (LANDA, 2016):

Equals

Objekty *a* a *b* jsou prostorově shodné, jestliže platí

$$a \subseteq b \wedge b \subseteq a$$

Disjoint

Objekty *a* a *b* jsou prostorově různé, jestliže platí

$$a \cap b = \emptyset$$

Intersects

Objekt *a* prostorově protíná objekt *b*, jestliže platí

$$a.Intersects(b) \Leftrightarrow !\,a.Disjoint(b)$$

Touches

Objekty *a* a *b* se prostorově dotýkají, jestliže platí

$$(I(a) \cap i(b) = \emptyset) \wedge (a \cap b \neq \emptyset)$$

Crosses

Prostorový vztah „křížení“ je definován pro bod/linie, bod/plocha, linie/linie a linie/plocha, jestliže platí

$$\dim\big(I(a) \cap I(b)\big) < \max\Big(dim\big(I(a)\big), dom\big(I(b)\big)\Big) \wedge (a \cap b \neq a) \wedge (a \cap b \neq b)$$

Within

Prostorový vztah „uvnitř“ je definován jako

$$(a \cap b = a) \wedge (I(a) \cap E(b) = \emptyset)$$

Contains

Objekt *a* obsahuje objekt *b*, jestliže platí

$$a.Contains(b) \Leftrightarrow b.Within(a)$$

Overlaps

Prostorový vztah „překrytí“ je definován pro plocha/plocha, linie/linie a bod/bod

$$\dim\big(I(a)\big) = \dim\big(I(b)\big) = \dim\big(I(a) \cap I(b)\big) \wedge (a \cap b \neq a) \wedge (a \cap b \neq b)$$

2. MĚŘENÍ VÝKONNOSTI

Ve světě se tomuto termínu říká „benchmarking“ a myslí se tím měření výkonnosti systému nebo části systému (podsystému) pomocí programu, souboru programů nebo operací. Nejčastější použití je u hodnocení výkonu různých komponent hardware (např. měření výkonnosti CPU). Benchmarkování ovšem lze použít i na software, typickým příkladem využití benchmarku na software je měření výkonnosti databázových systémů. Na trhu existuje mnoho benchmarků, které se užívají v mnoha různých oblastech, bohužel žádný z nich není uznaný za standard, ovšem ty nejlepší se k němu blíží, například transakční testy propustnosti TPC. (HUPPLER, 2009)

Většina prací, které se zabývají touto problematikou, jsou několik let staré a s vývojem inovací informačních technologií již můžeme tvrdit, že jsou neaktuální k současným trendům databází (např. NoSQL databáze nebo objektové databáze).

Bakalářská práce autora Miroslava Pazdery (PAZDERA, 2007) se zabývá konkrétním porovnáním dvou databázových řešení nad konkrétními daty. První databázové řešení je relační databáze MSSQL 2005 a druhým databázovým systémem je objektová databáze Caché. Porovnávání probíhá pomocí měření času několika jednoduchých SQL příkazů.

Lukáš Košárek se zabývá bakalářskou prací na téma „Výkonnostní srovnání relačních databází“ (KOŠÁREK, 2010). V této práci porovnává pouze několik různých SŘBD relačních databází. Testování probíhá pomocí vytvořené aplikace, která simuluje připojení více uživatelů najednou a měří čas prováděných operací.

Diplomová práce Tomáše Vrbíka (VRBÍK, 2012) se zabývá srovnáváním distribuovaných „NoSQL“ databází s důrazem na výkon a škálovatelnost. Tato práce z Vysoké školy ekonomické v Praze konkrétně porovnává 4 vybrané NoSQL databáze (MongoDB, Apache Cassandra, Apache HBase a Redis). Z důvodu, že se jedná o porovnání distribuovaných databází, je výkon srovnáván s využitím simulované zátěže v prostředí čtyřčlenného clusteru.

Bakalářská práce publikovaná Janem Virtem (VIRT, 2010) je spíše teoretického charakteru. Práce se zabývá tvorbou testu pro měření výkonnosti transakčních zpracování a následným zkoušením vytvořeného testu na firemní databázi. V práci jsou popsané požadavky na tvorbu úspěšného testu k měření výkonnosti databázového systému a testy neziskových organizací TPC a SPEC.

Diplomová práce studentky Elizabeth G. Reid (REID, 2009) řeší problém benchmarkování paměťových databází. Tato práce z Massachusetského institutu technologií navrhuje různorodou sbírku testů výkonnosti pro paměťové databáze. Součástí práce je i popis cílů testů, API pro práci s databází a testovací prostředí.

Další prací je výkonnostní porovnání prostorovo-časové databáze od Paula Wersteina (WERSTEIN, 1998), tato práce se zabývá porovnáním výkonu databází pracující s prostorovo-časovou složkou dat v 3–dimenzionálním prostoru. Celá práce se opírá o benchmark Sequoia 2000, který se skládá z jedenácti dotazů, které jsou podrobně popsány v textu (STONEBRAKER a kol., 1993).

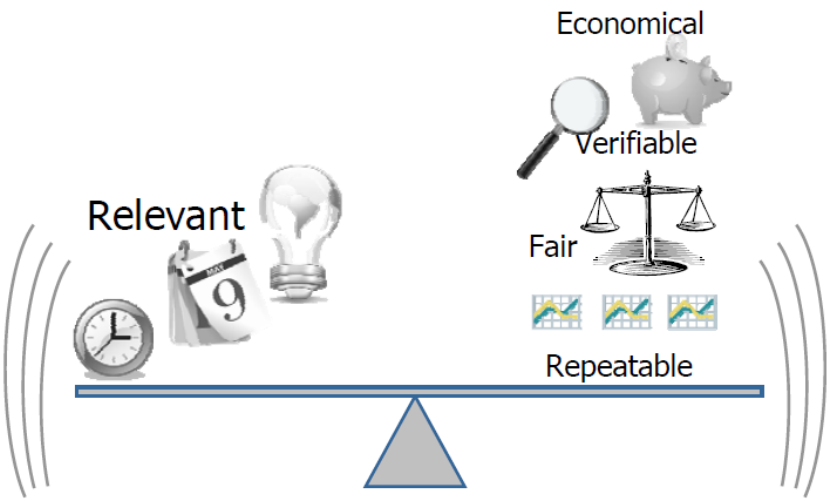
Práce (STONEBRAKER a kol., 1993) popisuje databázový test výkonnosti SEQUOIA 2000 aplikovaný na databáze ARC-INFO, GRASS, IPW a Postgres. V úvodu práce autor uvádí historii měření výkonnosti a stručně popisuje testy propustnosti TPC. V další části popisuje data, která jsou použita pro testování a seznámení čtenáře s testovacím scénářem, v poslední části jsou uvedené výsledky testu. V práci jsou přehledně popsané potřeby databázových systémů pro práci s prostorovými daty, testovací scénář je tvořen z 11 dotazů, které jsou rozdělené do kategorií „načítání dat“, „rastrové dotazy“, „bodové a polygonové dotazy“, „prostorové spojení“ a „rekurze“. Práce je ovšem 24 let stará a testovací scénář nepokrývá všechny oblasti zpracovávání prostorových dat, např. překryvné operace.

Navrhnout a následně implementovat benchmark je velice obtížný proces, který se řídí pěti základními pravidly. (HUPPLER, 2009)

- Relevantní – čtenář výsledků musí věřit, že se měří něco skutečného.
- Opakovatelný – výsledky při opakování testů musí být stejné.
- Spravedlivý – všichni hardware a software dodavatelé musí mít stejné

podmínky.

- Ověřitelný – musí být zaručeno, že výsledky testu jsou skutečné a že nedochází ke zkreslení údajů.
- Ekonomický – měření výkonnosti nesmí být příliš drahé.



Obr. 1 - Vlastnosti benchmarků. Zdroj (HUPPLER, 2009)

Pro dosažení optimálního poměru mezi všemi pravidly je často potřeba slevit z relevance.

2.1. Relevantní

Existuje několik charakteristik, které dělají měření výkonnosti relevantní nebo naopak nerelevantní:

- Smysluplná a pochopitelná jednotka.
- Zatížení softwarových prvků podobně jako u zákaznických aplikací.
- Zatížení hardwarových prvků podobně jako u zákaznických aplikací.
- Dlouhověkost – test výkonnosti by měl vydržet ideální dobu, ani moc krátkou, ani moc dlouhou.
- Široce využitelný.
- Mít cílovou skupinu, která bude chtít tyto informace.

I pro nezávislého čtenáře není problém pochopit charakteristiku smysluplné jednotky. Příkladem je jednotka „*SPECjbb bops*“ benchmarku SPECjbb_2005, zkratka „*ops“* z této jednotky znamená „operation per second“ „*b*“ je zkratkou „business“ a „jbb“ vyjadřuje „java business benchmark“, SPECjbb test se tedy prezentuje v počtu obchodních operací za sekundu v java business benchmarku. (HUPPLER, 2009)

Charakteristika využívání softwarových prvků jako u zákaznických aplikací je jedna z nejdůležitějších povah dobrého benchmarku, ale zároveň jedna z nejtěžších, co se týče konfigurace. Se stále se rozvíjejícím hardwarem i softwarem je obtížné navrhnout specifický benchmark (např. prostorový nebo decission support benchmark) tak, aby plnohodnotně využíval správné aplikační komponenty. Jako příklad povedeného benchmarku můžeme zmínit TPC-C, který převážně v 90. letech minulého století věrohodně simuloval transakční zpracování a sloužil pro mnoho společností k ladění vlastních produkčních aplikací. (HUPPLER, 2009)

Dlouhověkost je dalším důležitým aspektem, protože benchmark, který má životnost pouhý rok, není benchmarkem. Koncept softwaru, na který se má aplikovat benchmark, by měl být dostatečně moderní, ale na druhou stranu nesmí být příliš nový, aby neprocházel počátečním rychlým laděním. Příkladem v praxi může být zase SPEC, který své testy měření výkonnosti značí čísly verze a diskuze nad novou verzí začíná okamžikem vydání verze předchozí. (HUPPLER, 2009)

2.2. Opakovatelný

Tento požadavek na dobrý test výkonnosti zní jednoduše, ovšem v praxi je velice obtížný. Cílem tohoto požadavku je dosáhnout stále stejných výsledků při stejných vstupech. Bohužel při změně dat v databázové aplikaci se mění indexy a díky nim se může měnit query plány či tyto plány procházet jinými počty řádků. Hardwarové komponenty se také vždy nechovají stejně, disky se plní a fragmentují, a tudíž se chovají různě nebo například SSD disky mají seek time při zápisu na prázdné místo nižší než při přepisu. V TPC-C se tento problém řeší pomocí housekeeping transakcí, které během dvanácti-hodinového běhu testu udržují databázi v dobrém stavu, po uplynutí této doby je potřeba databázi obnovit. (HUPPLER, 2009)

2.3. Spravedlivý

Další z požadavků, který se na první pohled zdá jasný, ale i zde jsou určité problémy se zajištěním dodržování této charakteristiky. Současné databázové systémy mají své silné a slabé stránky, není tedy možné dodržet spravedlivost běhu v obecném prostředí. Existují ovšem kroky, které potenciální křivdu eliminují. Jeden z přístupu eliminace nespravedlivosti může být použití základních funkcí, které už všichni výrobci vyladili. Tento přístup ovšem dělá benchmark zastaralým již v době vydání. Důležité je si uvědomit, že každý produkt má své silné a slabé stránky výkonnosti. Dalším přístupem k eliminaci nespravedlivosti jsou různé omezení. Například benchmark *SPEC Java/ClientServer* je omezený pouze na Unixové systémy. (HUPPLER, 2009)

2.4. Ověřitelný

Aby nedocházelo k upravování výsledků pro vlastní prospěch databázovými vývojáři, musí být výsledky testů ověřitelné. Některé testy mají kontroly ověřitelnosti zabudované přímo ve své proceduře. Jiné benchmarky své výsledky zase ověřují nezávislým certifikovaným auditorem, takovým příkladem jsou testy TPC, dalším způsobem ověřitelnosti může být kontrola dobrovolníky. (HUPPLER, 2009)

2.5. Ekonomický

Vlastnost, která se při modelování testu často přehlíží. Důraz při modelování se klade více na charakteristiku realističnosti. Složité benchmarky často bývají drahé, protože vyžadují rozsáhlé testovací prostředí. Ekonomický nemá znamenat levný, ale má se jednat o dobrý poměr cena/výkon. Příkladem může být výsledek IBM v TPC-C, který činil 6 085 166 TpmC a vyžadoval zapojení 11000disků a 128 middle tear systémů. 1. místo v TPC-C bylo náležitě zveřejněno a IBM se investice vrátila formou zvýšených uzavřených kontraktů. (HUPPLER, 2009)

Obecně se mezi nejlepší a nejuznávanější databázové benchmarky řadí testy organizací TPC a SPEC.

2.6. TPC – Transaction Processing Performance Council

Nezisková organizace vytvořená za účelem vytvoření transakčních benchmarků nezávislých na výrobcích databází nebo hardwaru. V počátku měl TPC 8 členů, nyní jsou členy TPC všechny velké databázové a hardwarové firmy. Prvním benchmarkem, který TPC vydalo, byl TPC-A. Tento benchmark vycházel zejména z TP1 (předchůdce TPC firmy IBM). TPC-A pouze formalizoval běh a přidával auditing a proto vývoj a snaha o zlepšení testování výkonu databází vedla k vydání dalších verzí a druhů benchmarků. Mezi řadou dalších testů byl nejpopulárnější, a ještě nyní používaný TPC-C (1992). Benchmark zaměřený zejména na datové sklady byl vydán v roce 1995 pod názvem TPC-D – decission support benchmark, který byl později nahrazen TPC-H a TPC-R ve warehouse oblastech. TPC-E by měl být náhradníkem TPC-C. TPC-W a TPC-app měly jen krátkou životnost. (SHANLEY, K., 1998)

2.6.1. TPC-C

Tento online transaction processing benchmark byl po dvouletém vývoji poprvé představen v roce 1992. TPC-C obsahuje komplexní databázový model a více typů transakcí (5 různých souběžných typů). Transakce jsou jednoduché DML operace, spuštěny buď v rámci online zpracování nebo jsou nafrontovány k postupnému dávkovému zpracování. Databázový model je složen z 9 tabulek s rozdílným množstvím záznamů, různou strukturou a různými datovými typy. Jednotkou měření je tpmcC (TPC-C transakcí za minutu). (TPC-C, 1997)

Tento test simuluje prostředí, kdy skupina uživatelů pracuje s databází pomocí transakcí typu objednávka/vklad. Mezi tyto transakce patří zadávání objednávek, zaznamenávání provedených plateb, ověřování stavu objednávek a sledování množství zásob na skladu. Tento benchmark je vytvořen jako obraz velkoobchodu, ale není limitován jen na tento druh obchodní činnosti. Rozhodně ale není ideální pro porovnávání výkonu databází pracujících s prostorovými daty. (TPC-C, 1997)

2.7.SPEC – Standard Performance Evaluation Council

Nezisková organizace založena v roce 1988 s názvem *The System Performance Evaluation Cooperative*, později přejmenovaná na *Standard Performace Evaluation Council*. Tato organizace byla vytvořená několika výrobci pro potřebu realistického standardizovaného nástroje na měření výkonnosti systému. Základní myšlenkou organizace je, že trh potřebuje validní informace, a nejen marketingové produkty. Cílem organizace je vytváření opravdových aplikací pro benchmarkování namísto syntetických jednoduchých programů. (SPEC, 2009).

V současné době je SPEC tvořen 3 skupinami s rozdílnou působnosti:

- OSG (Open System Group) - původní organizace, která vede projektové sub týmy pracujících na benchmarcích pro měření výkonu.
- HPG (High Performance Group) – tato skupina si klade za cíl vytvářet benchmarky pro velice výkonné počítačové systémy (symetrické multiprocesorové systémy, clustery pracovních stanic, paralelní systémy s distribuovanou pamětí nebo tradiční vektorové a vektorově paralelní superpočítače).
- GWPG (Graphic and Workstation Performance Group) – skupina zastřešující vývoj konzistentních benchmarků pro porovnávání výkonnosti pracovních stanic a jejich grafických subsystému.

3. DATOVÝ MODEL

V této kapitole jsou popsána data organizace OpenStreetMap a nástroje, které s těmito daty pracují v diplomové práci. Tyto nástroje slouží k importu a následné migrací dat do databází.

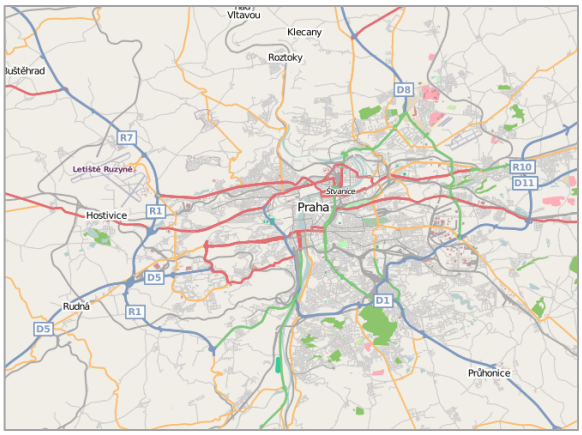
3.1. Open street maps

OpenStreetMap (OSM) je kolaborativní projekt založený v roce 2004, který má za cíl tvorbu volně dostupné, uživatelem editovatelné a webově přístupné mapy světa. Tento projekt je inspirován podobnými volně dostupnými projekty, například Wikipedií. Data jsou poskytovaná pod licencí *Open Database Licence* a lze je volně využívat i pro komerční účely. (WILLIS, 2007), (OpenStreetMap – Copyright, 2017)

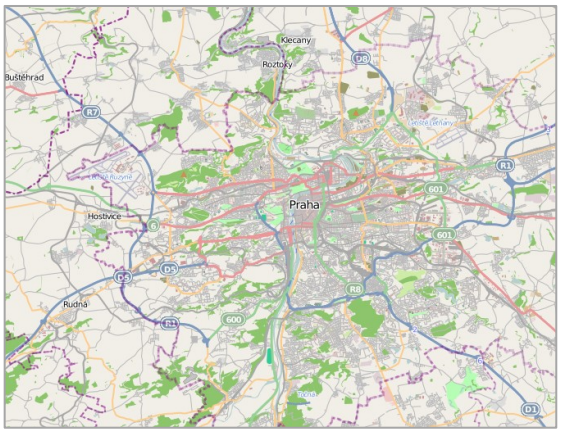
Projekt byl založený v červenci roku 2004 Stevem Coastem ve Velké Británii. O necelé 2 roky později byla založená stejnojmenná nezisková organizace s cílem povzbudit vytváření, zpracování a šíření geografických dat volně bez jakéhokoliv omezení. V prosinci 2006 společnost *Yahoo* poskytla své letecké snímky pro snazší tvorbu map a o 4 měsíce později poskytla společnost *Automotive Navigation Data* silniční mapy Nizozemska, Číny a Indie. Od té doby je mezi přispěvateli velké množství velkých společností (např. *Google*) a celý projekt má více než 1 000 000 registrovaných uživatelů. (OpenStreetMap, 2017)

Data OSM se používají v celé řadě webových portálů, nejznámější jsou portály *OpenStreetMap*, *CloudMade* nebo Čechy často využívané *Mapy.cz* (SLÍŽEK, 2014), které pomocí OSM vykreslují celosvětové území mimo ČR. Pro samotné přispívaní slouží mnoho nástrojů kresby a editačních nástrojů založených na různých technologiích (webové aplikace, desktopové, aplikace pro PDA či různé editory). Důležitými nástroji jsou *routing* neboli navigace, díky kterým lze pomocí OSM dat vytvářet jednoduché, zdarma navigační aplikace. Za zmínku stojí iniciativa *Humanitarian OpenStreetMap Team* (HOT), která má za cíl tvorbu map potřebných pro zasahující dobrovolníky při humanitárních krizích. Tato iniciativa společně s dalšími organizacemi založila roku 2014 projekt *Missing Maps*, který cílí na tvorbu map v nejvíce zranitelných místech světa. (WILLIS, 2007)

Data se dají stáhnout v několika formátech, za jednotlivé období a za určité území. Nejčastěji jsou data OSM ke stažení ve vlastním souborovém formátu postaveném na syntaxi XML. Referenční souřadnicové systémy používá projekt WGS 84 a pro rychlé a jednoduché zobrazení dat Mercatorovou projekci. (Geofabrik, 2016)



Obr. 2 - OpenStreetMap Prahy, duben 2008. Zdroj (OpenStreetMap, 2017).



Obr. 3 - OpenStreetMap Prahy, srpen 2009. Zdroj (OpenStreetMap, 2017).

3.1.1. Imposm

Imposm je nástroj pro importování dat OSM do PostgreSQL databáze. Tento nástroj je vyvíjený a podporovaný firmou Omniscale a lze jej pustit pod operačními systémy Linux a Mac OS X. Tento software je volně šiřitelný pod licencí Apache Software Licence 2.0. Nástroj vytváří v databázi tabulky, indexy a další užitečné databázové funkce pro snazší renderování dlaždic nebo podpory WMS. Práce s tímto programem probíhá v konzoli daného operačního systému. Import dat je rozdělený do dvou částí, v první části se data načítají do vlastních kešovacích souborů. (Imposm, 2016)

```
imposm --read germany-170101.osm.pbf
```

V druhé části se již načtená data nahrávají do databáze.

```
imposm --write --database osm --host localhost --user root
port 5432
```

3.1.2. GDAL – Geospatial Data Abstraction Library

Knihovna pro rastrová prostorová data dostupná s volně šiřitelnou licencí X/MIT pod hlavičkou nadace Open Source Geospatial Foundation. Pro práci s vektorovými daty slouží knihovna OGR, která je od verze 2.0 součástí knihovny GDAL. Knihovna je napsaná v programovacím jazyce C++. Jádro této knihovny poskytuje datovou abstrakci a rozhraní pro práci s daty. Knihovna obsahuje řadu nástrojů pro konverzi a zpracování dat. Práce s touto knihovnou probíhá skrze konzoli OS. (GDAL, 2017)

```
ogr2ogr --config PG_LIST_ALL_TABLES YES --config
PG_SKIP_VIEWS YES -f "SQLite" osm.sqlite -progress
PG:"dbname='osm' active_schema=public schemas=public
host='localhost' port='5432' user='root' password='XXX'" -lco
LAUNDER=yes -dsco SPATIALITE=yes -lco SPATIAL_INDEX=yes -gt
65536
```

4. METODIKA MĚŘENÍ VÝKONNOSTI PROSTOROVÝCH DATABÁZÍ

Tato kapitola popisuje celkový návrh testovacího scénáře, který slouží jako metodika k porovnávání výkonu databází nad daty OSM. Pro kvalitní a plnohodnotnou metodiku je potřeba definovat vhodný soubor dat, který je součástí kapitoly. Před samotným návrhem metodického scénáře pro testování výkonu databází je nezbytné si definovat požadavky, které má skupina testů dodržovat.

4.1. Data

Pro plnohodnotné otestování vlastností a schopností databázových systémů je potřeba testovat funkce na dostatečně velkém zdroji dat. Pro praktické využití je vybraný zdrojový balíček dat OSM zobrazující území Německa generovaný k datu 1. 1. 2017. Tento balíček dat je poskytován v komprimovaném formátu „*.pbf*“ a samotná velikost komprimovaného balíčku je 3,1 GB. Datový balíček lze stáhnout zdarma od poskytovatele OSM (Geofabrik, 2016). Datový zdroj použitý v práci je dostatečně velký pro věrohodné otestování výkonnosti databázového systému. Z důvodu dostupnosti a volného stažení datového zdroje, který se již nebude měnit, je tento balíček dat vhodný pro opakovatelný a spravedlivý test.

Z důvodu různých výsledků importu dat editačními nástroji pro import OSM/PBF souboru do databází je zvolen jeden nástroj pro import dat do PostgreSQL databáze a další nástroj pro import veškerých dat z jedné databáze do dalších. Tímto postupem je zaručeno, že model i samotná data ve všech testovaných databázích jsou totožná. Pro import surových dat do databáze je vybrán Imposm 2 software německé společnosti Omniscale GmbH & Co. KG. Pro přenos datového modelu a samotných dat je použita knihovna pro práci s prostorovými daty GDAL.

4.1.1. Datový model

Datový model je reprezentací krajinných prvků téměř totožný s mapovým klíčem standardní služby OSM, model je rozdělený do 14 tabulek, každá tabulka obsahuje sloupec s geometrií (PostgreSQL a SQLite – geometry, MySQL – SHAPE), k daným sloupcům s geometrií je vytvořený i prostorový index a k daným tabulkám je vytvořený primární klíč. Vzhledem k faktu, že nástroj slouží pro následné snazší renderování dlaždic nebo WMS

služeb, vytváří nástroj i generované tabulky, které ovšem nesou určitou duplicitu dat, a proto jsou tyto tabulky smazané. Po importu OSM dat tabulky a indexy dosahují velikosti 12 GB dat. Pro totožný datový model lze využít projekt (JACKOLIN, 2016), který umožňuje konverzi datového schématu importu programu *osm2pgsql* do datového schématu programu imposm.

Administrativní hranice

Vrstva administrativních hranic, která obsahuje polygonovou vrstvu správních jednotek. Tyto hranice jsou dělené podle atributu „*admin_level*“ podle důležitosti od národních států až po malé správní okrsky a předměstí. (Wiki OSM, 2014)

Tab. 1 - Vrstva administrativních hranic

Název tabulky:	**osm_new_admin**
Typ geometrie:	Polygon
Velikost tabulky:	44 MB
Počet záznamů:	441
Primární klíč:	osm_new_admin_pkey
Prostorový index:	osm_new_admin_geom
Atributy:	• id [integer] • osm_id [bigint] • name [character (255)] • type [character (255)] • admin_level [smallint] • geometry [geometry]

Letecké cesty

Tato vrstva se skládá z linií ranvejí a linií dopravních cest na letišti.

Tab. 2 - Vrstva leteckých cest

Název tabulky:	**osm_new_aeroways**
Typ geometrie:	Linie
Velikost tabulky:	1,27 MB
Počet záznamů:	6 555
Primární klíč:	osm_new_aeroways_pkey
Prostorový index:	osm_new_aeroways_geom
Atributy:	• id [integer] • osm_id [bigint] • name [character (255)] • type [character (255)] • geometry [geometry]

Veřejné zařízení

Bodová vrstva zobrazující univerzity, školy, knihovny, čerpací stanice, nemocnice, hasičské a policejní stanice nebo radnice.

Tab. 3 - Vrstva veřejných zařízení

Název tabulky:	**osm_new_amenities**
Typ geometrie:	Bod
Velikost tabulky:	4,18 MB
Počet záznamů:	41 204
Primární klíč:	osm_new_amenities_pkey
Prostorový index:	osm_new_amenities_geom
Atributy:	• id [integer] • osm_id [bigint] • name [character (255)] • type [character (255)] • geometry [geometry]

Budovy

Jedná se o největší tabulku v databázi. Polygonová vrstva půdorysů domů, budov, hal a opuštěných domů.

Tab. 4 - Vrstva budov

Název tabulky:	**osm_new_buildings**
Typ geometrie:	Polygon
Velikost tabulky:	4813 MB
Počet záznamů:	24 868 748
Primární klíč:	osm_new_buildings_pkey
Prostorový index:	osm_new_buildings_geom
Atributy:	• id [integer] • osm_id [bigint] • name [character (255)] • type [character (255)] • geometry [geometry]

Využití krajiny

Polygonová vrstva využití krajiny. Příklady těchto vrstev jsou lesy, traviny, zahrady, parkoviště nebo rezidenční a rekreační oblasti.

Tab. 5 - Vrstva využití krajiny

Název tabulky:	**osm_new_landusages**
Typ geometrie:	Polygon
Velikost tabulky:	1550 MB
Počet záznamů:	3 577 349
Primární klíč:	osm_new_landusages_pkey
Prostorový index:	osm_new_landusages_geom
Atributy:	• id [integer] • osm_id [bigint] • name [character (255)] • type [character (255)] • geometry [geometry]

Hlavní cesty

Tato liniová vrstva obsahuje veškeré pozemní komunikace první, druhé a třetí třídy.

Tab. 6 - Vrstva hlavních cest

Název tabulky:	**osm_new_mainroads**
Typ geometrie:	Linie
Velikost tabulky:	193 MB
Počet záznamů:	831 758
Primární klíč:	osm_new_mainroads_pkey
Prostorový index:	osm_new_mainroads_geom
Atributy:	• id [integer] • osm_id [bigint] • name [character (255)] • type [character (255)] • bridge [integer] • oneway [integer] • z_order [integer] • geometry [geometry]

Vedlejší komunikace

Vrstva obsahující linie pěšin, chodníku, cyklostezek, rezidenčních cest.

Tab. 7 - Vrstva vedlejších komunikací

Název tabulky:	**osm_new_minorroads**
Typ geometrie:	Linie
Velikost tabulky:	1786 MB
Počet záznamů:	8 668 562
Primární klíč:	osm_new_minorroads_pkey
Prostorový index:	osm_new_minorroads_geom
Atributy:	• id [integer] • osm_id [bigint] • name [character (255)] • type [character (255)] • bridge [integer] • oneway [integer] • z_order [integer] • geometry [geometry]

Dálnice

Vrstva zobrazující veškeré linie dálnic a silnic pro motorová vozidla.

Tab. 8 - Vrstva dálnic

Název tabulky:	**osm_new_motorways**
Typ geometrie:	Linie
Velikost tabulky:	27 MB
Počet záznamů:	140 611
Primární klíč:	osm_new_motorways_pkey
Prostorový index:	osm_new_motorways_geom
Atributy:	• id [integer] • osm_id [bigint] • name [character (255)] • type [character (255)] • bridge [integer] • oneway [integer] • z_order [integer] • geometry [geometry]

Popisky

Bodová vrstva popisků zemí, států, regionů, měst, obcí, osad a lokalit.

Tab. 9 - Vrstva popisků

Název tabulky:	**osm_new_places**
Typ geometrie:	Bod
Velikost tabulky:	12 MB
Počet záznamů:	127 842
Primární klíč:	osm_new_places_pkey
Prostorový index:	osm_new_places_geom
Atributy:	• id [integer] • osm_id [bigint] • name [character (255)] • type [character (255)] • z_order [integer] • population [integer] • geometry [geometry]

Železnice

Liniová vrstva železnic.

Tab. 10 -Vrstva železnic

Název tabulky:	**osm_new_railways**
Typ geometrie:	Linie
Velikost tabulky:	46 MB
Počet záznamů:	218 898
Primární klíč:	osm_new_railways_pkey
Prostorový index:	osm_new_railways_geom
Atributy:	• id [integer] • osm_id [bigint] • name [character (255)] • type [character (255)] • bridge [integer] • oneway [integer] • z_order [integer] • geometry [geometry]

Transportní plochy

Vrstva transportních ploch se skládá z vlakových nástupišť, leteckých terminálů a heliportů.

Tab. 11 - Vrstva transportních ploch

Název tabulky:	**osm_new_transport_areas**
Typ geometrie:	Polygon
Velikost tabulky:	1,24 MB
Počet záznamů:	2975
Primární klíč:	osm_new_transport_areas_pkey
Prostorový index:	osm_new_transport_areas_geom
Atributy:	• id [integer] • osm_id [bigint] • name [character (255)] • type [character (255)] • geometry [geometry]

Transportní body

Bodová vrstva zastávek, stanic, přechodů, podchodů, leteckých terminálů, heliportů a vstupních bran.

Tab. 12 - Vrstva transportních bodů

Název tabulky:	**osm_new_transport_points**
Typ geometrie:	Bod
Velikost tabulky:	47 MB
Počet záznamů:	507 113
Primární klíč:	osm_new_transport_points_pkey
Prostorový index:	osm_new_transport_points_geom
Atributy:	• id [integer] • osm_id [bigint] • name [character (255)] • type [character (255)] • geometry [geometry]

Vodní plochy

Vrstva vodních ploch.

Tab. 13 - Vrstva vodních ploch

Název tabulky:	**osm_new_waterareas**
Typ geometrie:	Linie
Velikost tabulky:	123 MB
Počet záznamů:	307 400
Primární klíč:	osm_new_waterareas_pkey
Prostorový index:	osm_new_waterareas_geom
Atributy:	• id [integer] • osm_id [bigint] • name [character (255)] • type [character (255)] • geometry [geometry]

Vodní toky

Vrstva řek, potoků a kanálů.

Tab. 14 - Vrstva vodních toků.

Název tabulky:	**osm_new_waterways**
Typ geometrie:	Linie
Velikost tabulky:	115 MB
Počet záznamů:	218 898
Primární klíč:	osm_new_waterways_pkey
Prostorový index:	osm_new_waterways_geom
Atributy:	• id [integer] • osm_id [bigint] • name [character (255)] • type [character (255)] • geometry [geometry]

4.2. Požadavky na testovací scénář

Kvalitní test pro měření výkonností se musí řídit pěti základními pravidly z kapitoly 2. Některá pravidla je ovšem velice obtížné dodržovat. Například charakteristika ověřitelnosti každého testu pomocí nezávislých kontrolerů třetích stran.

Testovací scénář musí popisovat víceúčelovost databázových systémů, proto v testovacím scénáři nejsou zahrnuté jenom prostorové dotazy, ale i dotazy nad obecnými tabulkovými daty.

Různé prostorové databáze mají odlišné typy operací s prostorovými daty, a proto je velice obtížné porovnat tyto databáze mezi sebou. Ovšem většina prostorových databází prvořadě podporují standardizované funkce, a proto dané funkce, které jsou implementované v testovacím scénáři, musí podléhat standardizaci organizace Open Geospatial Consortium (OGC). Důvodem tohoto požadavku je, že většina databázových systémů, které umožňují práci s prostorovými daty, mají primárně implementované tyto funkce a cílem této metodiky je využitelnost těchto testů na co největší množství databázových systémů podporující práci s prostorovými daty.

Testovací scénář musí umožňovat testování nejenom relačních transakčních databází, ale musí být schopný implementovat celou řadu moderních druhů databázových systémů (objektové databáze, NoSQL databáze). Sada testů může dobře sloužit k testování vlastní implementace prostorového rozšíření objektových databázových systémů, příkladem takového objektového rozšíření lze vidět v práci (POLÁCH, 2012).

Posledním požadavkem je otevřenost. Tímto požadavkem se myslí, že veškeré parametry daného testu, dotazů a testovacích prostředí budou komukoliv volně dostupné. Tímto požadavkem je dosaženo maximální ověřitelnosti a relevantnosti měření výkonnosti.

4.3. Testovací scénář

Testovací scénář se skládá z komplexních dotazů, které se pokouší široce popsat víceúčelovost daných databází. Tato sada testů se skládá ze 4 logicky uspořádaných celků, které jsou dále podrobněji popsané. Oproti jiným pracím, tato práce neobsahuje databázové dotazy, které vytvářejí dané tabulky a hierarchii datového modelu (SQL příkazy pro definici dat), tyto dotazy uživatel nebo administrátor databáze využívá minimálně (ve většině případů jen jednou) a pro běžné použití nehraje žádnou významnou roli rychlost těchto dotazů. Daný testovací scénář netestuje paralelní přístup a transakční zpracování. Na toto téma existuje větší množství prací, které se zabývají porovnáním databázových systémů z hlediska přístupu více uživatelů zároveň, příklady takových prací jsou (PAZDERA, 2007), (KOŠÁREK, 2010), (VIRT, 2010) nebo testy organizací (TPC – C, 1997), (SPEC, 2009).

Oproti jiným případům (např. měření výkonnosti transakčních zpracování), které používají k prezentaci svých výsledků většinou jednotku založenou na počtu transakcí za určitou dobu, nám postačí k prezentaci čas, který systém potřebuje k nalezení celkové množiny prvků. Přesnost měření se bude určovat na setinu sekundy a pro vyloučení náhodné chyby bude výpočet tohoto času průměr z deseti opakování.

Prostorové funkce se nejčastěji využívají k porozumění informacím a analýzám z dostupných dat, proto většina testů v diplomové práci slouží k získání informací nebo množiny dat.

4.3.1. Tematické dotazy

Do tohoto logického celku patří databázové dotazy, které nemají nic společného s prostorovou složkou dat. Mnoho případů využití prostorových databází se může dotazovat pouze na tematickou složku dat a geometrická složka je v těchto případech nepotřebná. Proto je důležité zaměřit své hodnocení daných databází i na neprostorovou složku dat. Do tohoto logického celku databázových testů patří 6 dotazů:

1. Najdi jedinečné hodnoty typu budov.
2. Najdi id a název vodních toků vzestupně setříděné podle názvu.
3. Najdi id a název vodních segmentů, které mají název „Elbe“.
4. Smaž všechny popisky, které mají typ „village“.
5. Vlož záznamy z hlavních cest a motorových cest do nové tabulky comunication.
6. Najdi všechny id tabulky comunication a id tabulky hlavních cest, které jsou společně přirozeně spojené s tabulkou comunication přes atributy osm_id.

4.3.2. Geometrické dotazy

Do této skupiny dotazů patří dotazy, které využívají prostorovou složku dat pro potřeby výpočtů geometrických vlastností. Tato kategorie se skládá z dotazů, které se zaměřují na vyhledávání podle určité geometrie a z dotazů výpočetně náročných. Cílem těchto dotazů je otestovat správnost funkcí a optimalizací výpočetních relací, které vypočítají z jednoho nebo více objektů nějakou hodnotu nebo vytvoří nový prvek.

7. Najdi všechny id a názvy škol z tabulky veřejných zařízení v MBR 52,5 s.š. 13,35 v.d. a 52,55 s.š. 13,4 v.d.
8. Najdi všechny id a názvy hotelů z tabulky budov do vzdálenosti 10 000 m z bodu 52,5 s.š. a 13,4 v.d.
9. Vypočti rozlohu vodních ploch.
10. Zapiš do nového sloupce délky všech segmentů silnic.
11. Vytvoř buffer leteckých cest velikosti 50 m.
12. Vypočti centroid transportních oblastí.
13. Vyber geometrii železnic transformovanou do formátu WKT.

4.3.3. Predikátové dotazy

Třetí část testovacího scénáře se skládá z predikátových dotazů. Skupina dotazů založených na prostorovém predikátu vrací výsledky v závislosti na další vrstvě a pomocí těchto funkcí dochází k prostorovému spojení. Naprostá většina analýz nad prostorovými daty se skládá z těchto dotazů a zpracování těchto dat umožňuje menší výpočetní náročnost aplikační vrstvy a menší zatížení sítě z důvodu menšího přenosu dat. Součástí této skupiny testů jsou také geometrické relace, které mají prostorový vztah mezi 2 prvky a výsledkem je jedna či více hodnot prostorového typu.

14. Najdi id, názvy a typy veřejných zařízení, které neleží ve spolkové zemi „Hamburg“.
15. Najdi id všech přítoků řeky „Elbe“.
16. Najdi id a název vodních ploch, které alespoň částí překrývají přírodní rezervace.
17. Najdi id a název vodních ploch, které celé leží v krajinném pokryvu typu „forest“.
18. Najdi id a název přírodních rezervací, které se shodují s krajinným pokryvem typu „grass“.
19. Najdi všechny oblasti, na kterých se překrývají plochy krajinného pokryvu typů „nature_reserve“ a „forest“.
20. Spoj geometrii bodů veřejných zařízení typu „university“ a „school“.
21. Najdi části hospodářských oblastí, které nezasahují do oblastí travin, ale současně se s nimi protínají.

4.3.4. Analytické dotazy

Poslední skupinou dotazů jsou výpočetně složité dotazy skládající se z několika různých jednodušších výpočtů. Těmito dotazy jsou standardní geoinformační funkce, se kterými se geoinformatik běžně setkává. Tato skupina dotazů testuje, zda je databázový systém vhodný k náročným výpočtům a zda testovaným databázovým systémem lze nahradit aplikační vrstvu architektury IS.

22. Vypočti hustotu dálnic v jednotlivých okresech.
23. Vypočti hustotu dálnic v přepočtu na obyvatele.
24. Vypočti rozlohu jednotlivých krajinných pokryvů v okrese „Worms“.
25. Vypočti míru ekologické stability krajiny v okrese „Worms“.

5. TESTOVÁNÍ

5.1. Výběr databází

Tato kapitola popisuje zdůvodnění výběru databází, které jsou použité v praktické části diplomové práce. V této kapitole jsou také popsané jednotlivé databázové systémy, jejich vlastností a prostorová rozšíření.

Pro praktickou část práce jsou zvolené 3 databázové systémy, které mají několik společných vlastností. Pro práci s daty je pro všechny 3 databázové systémy definovaný jazyk pro manipulaci s daty SQL. Tento jazyk se v různých případech liší, ale základní syntaxe je stejná. Dalším důležitým kritériem pro výběr databáze je svobodné licencování pro komerční užití. Na trhu existuje mnoho volně dostupných databázových řešení, jejich popularita mezi uživateli stále roste a funkcionalita i podpora těchto řešení se přibližuje nebo v některých případech překračuje placený software. Třetím kritériem pro výběr databázových řešení je podpora standardu Simple Features mezinárodní organizace Open Geospatial Consortium (OGC). Posledním kritériem pro výběr je obliba a použitelnost v praxi. Po zvážení těchto kritérií jsou zvolené databázové řešení PostgreSQL s prostorovou nadstavbou PostGIS, SQLite s nadstavbou Spatialite a MySQL.

Jeden z cílů použité metodiky porovnávání prostorových databází je, že daný testovací scénář musí být aplikovatelný i na jiné databáze než jen relační. Z tohoto důvodu je mezi výběrem databázových řešení SQLite, která sice je relační databáze, ale svou architekturou se od databází typu klient/server liší. SQLite má navíc široké uplatnění v mobilních aplikacích. Pro vhodnější reprezentaci tohoto cíle by bylo rozumné využít například NoSQL řešení CouchDB s prostorovou nadstavbou GeoCouch, která umožňuje implementaci prostorového klíče. Dané NoSQL řešení ovšem nemá podporu prostorových funkcí, a tudíž by se pro testování funkcí musel použít jeden z programovacích jazyků, který umožňuje práci s databázovým systémem.

Databázový systém PostgreSQL s prostorovou nadstavbou PostGIS je vybrán z důvodu široké podpory prostorových funkcí a veliké obliby v GIS komunitě uživatelů. Třetí databázový systém MySQL s prostorovým rozšířením je v práci zahrnut na základě veliké obliby ve webových technologiích. MySQL databáze v kombinaci s operačním systémem

Linux, webovým serverem Apache a programovacím jazykem na straně serveru PHP tvoří velice silnou sadu (LAMP) pro tvorbu webových aplikací.

5.1.1. PostgreSQL a Postgis

Volně šiřitelný a zdrojově otevřený objektově-relační systém vydávaný pod licencí *MIT*. PostgreSQL, někdy taky jednodušeji Postgres, je vyvíjen globální komunitou vývojářů primárně pro unixové systémy. PostgreSQL si zakládá na pověsti spolehlivého systému s dobrou integritou dat. (PostgreSQL about, 2016)

Historie

PostgreSQL se vyvinul ze systému Ingres, když prof. Michael Stonebraker stávající projekt v roce 1986 dále rozvíjel pod označením Postgres. Dalším milníkem bylo o několik let později přidání podpory SQL a v roce 1995 byla uvolněna verze pod názvem Postgres95. Široká obliba dotazovacího jazyka SQL pomohla systému mezi hlavní databázové produkty a v roce 1996 byl projekt s otevřeným zdrojovým kódem vypuštěn mezi veřejnost pod názvem PostgreSQL. (BLUM, 2007)

V letech 1991 a 1992 byly do PostgreSQL implementovány datové typy pro podporu GIS. Vývoj tohoto databázového systému je stále dynamický, nová verze systému vychází v ročních intervalech a opravy stávajících verzí jsou dostupné v čtvrtletních intervalech. (STĚHULE, 2012)

Vlastnosti

PostgreSQL obsahuje většinu datových typů ze standardů SQL92 a SQL99, má podporu ukládání binárních objektů, cizích klíčů, spojování tabulek, uložených procedur pohledu a trigerů. PostgreSQL splňuje podmínky pro bezpečný transakční systém, těmto podmínkám se říká ACID (Atomicita, Konzistence, Isolovanost a Trvalost). Systém je Case-sensitivní (záleží na velikosti písma) a rozšiřitelný, což znamená, že uživatelé jej mohou rozšiřovat například o vlastní funkce, datové typy nebo operátory. PostgreSQL databázový systém je velice škálovatelný ohledně množství dat, které může spravovat, ale i počtu uživatelů, kteří s ním mohou paralelně pracovat. (PostgreSQL documentation, 2017)

PostGIS

Jedná se o prostorovou nadstavbu databázového systému PostgreSQL, která umožňuje manipulaci a uchovávání prostorových dat. Tato nadstavba splňuje standardy OGC a společně s PostgreSQL se jedná o nejrozšířenější řešení na trhu v GIS. PostGIS přináší nové datové typy, operátory a mnoho funkcí rozdělených podle účelu (například geometrické konstruktory, doplňky, editory, výstupy, funkce prostorových spojení nebo měření), jejich celkový přehled je dostupný na (PostGIS reference, 2016). PostGIS umožňuje přidání prostorového indexu GIST. (PostGIS manual, 2016)

Práce s databází

Stejně jako většina rozšířených databázových nástrojů má i PostgreSQL vlastní grafické uživatelské rozhraní pro administraci databáze. Nazývá se *PgAdmin* a je volně dostupný z (PgAdmin, 2017), dalším nástrojem pro práci s PostgreSQL databází je *phpPgAdmin* (PhpPgAdmin, 2013), jak už z názvu vyplývá, jedná se o webovou aplikaci, která slouží k administraci databáze. Třetím nástrojem pro práci s databázemi PostgreSQL uvádím konzolovou aplikaci *psql* (PSQL, 2017), která je standardem při instalaci PostgreSQL serveru.

Výhody a nevýhody

Hlavní výhodou PostgreSQL je open-source licence, díky které je možné libovolně upravovat a rozšiřovat databázi i pro komerční účely. Tento systém komerčním systémům dělá konkurenci a v mnoha případech je i poráží. Další výhodou tohoto databázového systému je přehledná a obsáhlá dokumentace dostupná na internetu. (PostgreSQL about, 2016)

Mezi hlavní nevýhody můžeme označit nedostatečnou rozšiřitelnost na všech serverech poskytovaných webhostingy. Další nevýhodou je, že není podporované fulltextové vyhledávání přímo v PostgreSQL a je potřeba použít jednu z nadstaveb. (KYSILKA, 2003)

5.1.2. SQLite a Spatialite

Jedná se o databázový systém obsažený v menší knihovně napsaný v jazyce C. Tento systém je šířen pod licencí *public domain* a na rozdíl od databází založených na principu

klient-server, SQLite je databázový server spouštěn samostatným procesem. Každá databáze je ukládána v souboru s koncovkou „*.dbm*“. (SQLite About, 2016)

Vlastnosti

V SQLite systémech je implementovaný téměř celý standard SQL92, systém podporuje transakce, cizí klíče, pohledy (jen pro čtení), trigery (nepodporuje konstrukci „for each statement“) a spojování tabulek. SQLite má deklarovanou vlastnost „typeless“, takže do sloupce definovaný jako number můžeme uložit textový řetězec. (SQLite Documentation, 2016)

Spatialite

Jedná se o prostorové rozšíření SQLite databáze, které umožňuje uchovávání a manipulování s prostorovými daty. Data se ukládají do datového typu „Geometry“ a prostorové funkce jsou specifikované standardem OGC-SFS (OGC Simple Features). Celá implementace rozšíření je závislá na několika open-source knihovnách (LIBICONV, GEOS, PROJ4). Veškeré vlastnosti SQLite jsou implementované i v Spatialite. Kompletní přehled funkcí je popsaný v referenční příručce (SpatiaLite – Reference, 2015). Spatialite umožňuje aplikování prostorových indexů na tabulku, konkrétně algoritmus založený na konceptu R*tree (SILBERSCHATZ a kol., 2006), (SpatiaLite, 2013), (SpatiaLite – GIS Basic, 2015), (SpatiaLite – GIS Advanced, 2015)

Práce s databází

Pro práci s databázemi SQLite nebo Spatialite existuje poměrně mnoho uživatelsky přívětivých nástrojů. Typickým příkladem pro vytváření a úpravu databází SQLite je jednoduchá aplikace DB Browser, ke stažení je (SQLite browser, 2016). Na rozdíl od prostorových rozšíření databází MySQL a PostgreSQL, existuje grafické uživatelské rozhraní (GUI) pro nadstavbu SpatiaLite (SpatiaLite – GIS, 2017). Toto GUI umožňuje uživatelům snadnou editaci a vytváření geografických dat. Mimo tyto GUI lze použít pro práci s databázemi konzolovou aplikaci.

Výhody a nevýhody

Výhodou tohoto systému je jeho multiplatformita, ta se využije v případech, budeme-li potřebovat přenášet databázi mezi operačními systémy. Další výhodou systému je schopnost

šetřit systémové prostředky. (SQLite difference, 2016).

Nevýhodu SQLite má v absenci typové kontroly dat, díky kterým mohou v databázi vzniknout datové nekonzistence. Další nevýhodou může být fragmentace, která způsobuje neefektivitu práce s uloženými daty. Poslední nevýhodou tohoto systému je neschopnost českého třídění. (SQLite difference, 2016).

5.1.3. MySQL

Relační multiplatformní databázový model vytvořený švédskou firmou MySQL AB, v současnosti vlastněný Oracle, je k dispozici pod komerční placenou licencí (verze deluxe), tak i pod bezplatnou licencí GPL. Pro svůj vysoký výkon, snadnou implementaci a volně šiřitelnou licencí má velikou oblibu u webových technologií v kombinaci se serverem Apache a programovacím jazykem PHP. Hlavní optimalizace byla od počátku směřovaná na rychlost a další vlastnosti (např. zálohování, pohledy, triggery) nebyly příliš podporované. MySQL umožňuje ukládat data do několika typů databázových tabulek (storage engine), nejznámější a nejvíce používané jsou MyISAM a InnoDB. (GILFILIAN, 2003), (WELLING a THOMSON, 2005), (MySQL Engines, 2017)

MyISAM

Tabulky typu MyISAM jsou nástupcem tabulek typu ISAM. Hlavní nevýhodou těchto tabulek je, že nepodporují transakce. Při práci se záznamy v této tabulce se zamyká celá tabulka, tato strategie zamykání je na některé typy operací nevhodná. Tento typ tabulek na rozdíl od typu ISAM komprimuje indexy, tudíž zabírají tabulky mnohem méně místa. Typ tabulek MyISSAM na rozdíl od tabulek InnoDB umožňuje fulltextové vyhledávání a práce s těmito tabulkami je výrazně rychlejší. (MySQL Engines, 2017)

InnoDB

Tabulky typu InnoDB mají hlavní výhodu v provádění transakcí. Strategie zamykání je na úrovni řádků, a tudíž k jedné tabulce může paralelně přistupovat více uživatelů. Rozdíl oproti MyISAM je v reprezentaci tabulek a indexu. U MyISAM jsou objekty ukládány v určitém adresáři, u InnoDB jsou tabulky a indexy ukládány v prostoru tabulek, který je reprezentován jediným souborem. Obě tabulky umožňují definování sloupců, které by měly sloužit jako cizí klíč, ovšem pouze tabulky typu InnoDB provádí kontrolu existence

záznamů. (MySQL Engines, 2017)

Vlastnosti

MySQL je v bezplatné verzi multiplatformní, lze ji provozovat na 32bit i 64bit architekturách, umožňuje spouštět uložené procedury, triggery, databázové pohledy nebo práce s metadaty. Dále databázový server umožnuje podvýběry, práci s primárními klíči, indexy. V tabulkách typu InnoDB je podporovaná kontrola cizích klíčů a také databázový systém na těchto tabulkách umožňuje řízení transakcí. (MySQL Reference, 2017)

Prostorová nadstavba

Prostorové rozšíření databáze MySQL podporuje funkce specifikace Open Geospatial Consortium. Rozšíření umožňuje uchovávání a práci s daty skrze nejrůznější funkce. Do verze MySQL 5.0.16 byly tyto funkce dostupné pouze pro tabulky MyISAM, od této verze jsou prostorové funkce dostupné i pro další typy tabulek. Celkový přehled funkcí pro práci s prostorovými daty je setříděný v (MySQL Reference, 2017). MySQL umožňuje přidat prostorový index R-Tree.

Práce s databází

MySQL má na rozdíl od předešlých databázových řešení pro práci s databázemi klienta, který na trhu oproti ostatním nástrojům dominuje. Tento klient se nazývá phpMyAdmin a je předinstalovaný na mnoha českých i zahraničních webhostinzích (PhpMyAdmin, 2017). Dalším produktem je webová aplikace Adminer od českého webového vývojáře Jakuba Vrány (Adminer, 2016). Dalšími často používanými aplikacemi je HeidiSQL (HeidiSQL, 2017), která běží na platformě Windows a konzolová aplikace MySQL. (LACKO, 2011)

Výhody a nevýhody

Databázový systém MySQL je nenáročný, rychlý a spolehlivý. Existuje pro většinu platforem OS počítačů. Z důvodu optimalizace na výkon databázový systém nepodporuje složitější programátorské konstrukce a v náročných webových aplikacích nemá MySQL dostatečný výkon.

5.1.4. Použité verze a nastavení

K samotné práci jsou použité aktuální oficiální verze systémů dostupné z oficiálních zdrojů. Vzhledem ke skutečnosti, že primárním cílem práce není nalézt optimální konfiguraci databázového systému, ponechám databázovým systémům výchozí konfigurace. Nalezení optimální konfigurace databázového systému daleko převyšuje okruh této práce. Součástí výsledku měření výkonnosti databázových systémů jsou konfigurace použitých databázových systémů (Příloha 1, Příloha 2).

Aktuální verze systému k datu provádění testů:

- PostgreSQL 9.5.6
- PostGIS 2.0
- SQLite 3.11.0
- SpatiaLite 4.3.0
- MySQL 5.7.17

5.2. Příprava před testováním

Prvotní fází k přípravě testování jsou instalace vybraných databázových řešení, z důvodu práce v příkazové řádce operačního systému Linux jsou instalace otázkou několika minut. Před samotnou instalací a přípravou všech potřebných částí je vytvořený druhý testovací server. Na tomto zkušebním serveru se zkouší postupy instalací potřebných aplikací a průběh testů, aby testování v produkčním prostředí probíhalo bez jakýchkoliv problémů a přebytečných aplikací, které by mohly ovlivňovat výsledky testů. Testovací prostředí také slouží k ladění příkazů dotazů, aby dané SQL příkazy byly optimalizované. Kromě instalací databázových serverů je potřeba nainstalovat i nástroje k importu a práci s daty.

5.2.1. Import dat

Import dat se dělí na 2 etapy, první etapa se skládá z vytvoření tabulek a indexů v databázi PostgreSQL s prostorovou nadstavbou PostGIS pomocí importovacího nástroje Imposm. Ještě před samotným spuštěním nástroje je potřeba vytvořit prázdnou databázi a vytvořit prostorovou extenzi PostGIS, prostorová extenze se vytváří v konzoli OS pomocí příkazů:

```
psql -d osm -c "CREATE EXTENSION postgis;"
```

Dále se spouští importovací nástroj, který prvně nahrává OSM soubor do kešovacích souborů, ze kterých následně zapisuje do databáze. Import datového balíčku o velikosti 3,1 GB na relativně slabém stroji trvá okolo 27 hod. Importovací nástroj průběžně vypisuje informační hlášky z importu dat, díky kterým je jistota, že import našeho balíku dat probíhá v pořádku. Po úspěšném nahrání všech dat se odstraní generalizované tabulky příkazem pro smazání tabulky.

```
DROP TABLE název_tabulky CASCADE;
```

Konkrétně jde o tabulky:

- osm_new_landusages_gen0
- osm_new_landusages_gen1
- osm_new_mainroads_gen0
- osm_new_mainroads_gen1
- osm_new_motorways_gen0
- osm_new_motorways_gen1
- osm_new_railways_gen0
- osm_new_railways_gen1
- osm_new_waterareas_gen0
- osm_new_waterareas_gen1

Nyní se provádí kontrola validity geometrie dat, ta se provádí pomocí funkce *ST_IsValid*, která vrací pravdivostní hodnotu. Záznamy, které obsahují nevalidní geometrii, jsou smazány.

Druhou etapou importu je nahrávání dat z databáze PostgreSQL do databází SQLite a MySQL. Tento import se provádí z důvodu konzistence dat a totožného datového modelu pomocí GDAL knihovny, která dokáže importovat data z jedné databáze do druhé i s tvorbou datové struktury (tabulky, indexy). V případě databázového systému SQLite nástroj dokonce vytvoří prázdnou databázi. V případě MySQL je potřeba před spuštěním importu dat vytvořit

prázdnou databázi a změnit znakovou sadu databáze na *UTF8*.

```
ogr2ogr --config PG_LIST_ALL_TABLES YES --config
PG_SKIP_VIEWS YES -f "MySQL"
MYSQL:"osm,host=localhost,user=root,password=Abc253130,port=3
306" -progress PG:"dbname='osm' active_schema=public
schemas=public host='localhost' port='5432' user='root'
password='Abc253130'"
```

Posledním krokem importu je kontrola dat všech 3 databází. Kontrola spočívá v ověření, zda se nahrály všechny indexy k jednotlivým tabulkám, zda je v jednotlivých tabulkách pro každou databázi totožný počet záznamů a zda se v každé databázi správně zobrazuje znaková sada.

Poslední fází před samotným testováním je vytváření tabulek, sloupce a indexů pro otestování rychlosti příkazů *Insert* a *Update*. Vytvořené tabulky a sloupce jsou pro všechny 3 databáze totožné. Příkazy pro vytváření tabulek, sloupce a indexů jsou v každé databázi odlišné, například MySQL prostorové indexy řídí automaticky, ale u databáze SpatiaLite je potřeba vytvářet specifické spouštěče (triggery). Příklad tvorby tabulky v databázovém systému MySQL:

```
CREATE TABLE `comunication` (
     `id` int(11) NOT NULL AUTO_INCREMENT,
     `SHAPE` geometry NOT NULL,
     `osm_id` bigint(20) DEFAULT NULL,
     `name` varchar(255) DEFAULT NULL,
     `type` varchar(255) DEFAULT NULL,
     `tunnel` decimal(5,0) DEFAULT NULL,
     `bridge` decimal(5,0) DEFAULT NULL,
     `oneway` decimal(5,0) DEFAULT NULL,
     `length` FLOAT DEFAULT NULL,
     `z_order` decimal(5,0) DEFAULT NULL,
     UNIQUE KEY `id` (`id`),
     SPATIAL KEY `SHAPE` (`SHAPE`)
     ) ENGINE=InnoDB AUTO_INCREMENT=10588 DEFAULT
CHARSET=utf8;
```

Před samotným začátkem testování je vhodné spustit příkaz *Vacuum full*, který umožňuje odstranit neaktuální verze řádků z datových souborů a optimalizují místo na disku. (CRHONEK, 2011) Novější verze databázových systémů (například PostgreSQL od verze

7.4.3) už mají zabudovanou režii optimalizace s paměťovým prostorem ve vlastních rukách.

5.3. Testovací prostředí

Testování probíhá na virtuálním linuxovém stroji společnosti Internet CZ, a.s.. Tato společnost nabízí služby Cloud Computingu založeným na modelu Iaas (Infrastructure as a service). Virtuální servery je možné spravovat požitím VMware nebo HyperV hypervizorem s plnou škálovatelností pro CPU, RAM a diskový prostor. Pro tuto práci je vybraná služba Cloud server Smart, která umožňuje vytvořit server s až 4 procesory, 8 GB operační paměti a 160 GB diskového prostoru. Tento systém je postaven na technologii hypervizoru VMware s lokálním SSD úložištěm (Internet CZ, 2017). Levnější a méně výkonná služba Cloud server Smart byla vybraná z důvodu otestování navržené metodiky na slabším stroji a potvrzení, že dané testy je možné aplikovat i na stroji, které mají výkon srovnatelný s běžně používanými notebooky.

Testovací prostředí použité v této práci běží v nejvyšším možném módu služby Cloud server Smart. Tato služba po stránce výkonu nemůže konkurovat profesionálním databázovým serverům. Ale pro otestování a předvedení vytvořené metodiky je dostačující.

Operačním systémem pro diplomovou práci je zvolen *Ubuntu 16.04 Xenial Xerus* ve verzi *Server install image* s 64bitovou architekturou počítače. Tato varianta Ubuntu 16.04 neinstaluje uživatelské grafické rozhraní a převážně slouží jako server.

5.4. Průběh testování

Měření výkonnosti se odehrává v konzolových aplikacích konkrétního databázového systému. Každý test se opakoval 10x s uvolněním paměti operačního systému. Tento krok daným testům zaručí, že se k datům přistupuje z disku, a ne z paměti operačního systému, která značně ovlivňuje rychlost načítání dat a zpracování doby dotazu. Tento způsob dotazování je časově náročnější, ovšem rozdíly ve výpočtech databázových systémů jsou relevantní. Uvolňování paměti probíhá na linuxových strojích pomocí jednoduchého příkazu v konzolové aplikaci. Výřez obrazovky s konzolovou aplikací a práci v MySQL je na Obr. 4.

```
mysql> use osm;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> SELECT id, osm_id,  name, type, ST_AsText(SHAPE) as SHAPE  FROM osm_new_amenities LIMIT 5;
+----+----------+------+------+-----------------------------------------+
| id | osm_id   | name | type | SHAPE                                   |
+----+----------+------+------+-----------------------------------------+
|  1 | 16541597 | Aral | fuel | POINT(1485607.51842049 6899538.3913949)  |
|  2 | 16548176 | JET  | fuel | POINT(965640.355156844 6368275.31159265) |
|  3 | 16655326 | NULL | fuel | POINT(975179.422510462 6382444.84587443) |
|  4 | 16655329 | JET  | fuel | POINT(976886.184075255 6385426.77637066) |
|  5 | 17983325 | Aral | fuel | POINT(1304934.8048764 6124936.27461726)  |
+----+----------+------+------+-----------------------------------------+
5 rows in set (0.00 sec)
```

Obr. 4 - Práce v konzolové aplikaci s MySQL databázi. Zdroj vlastní.

5.5. Problémy při testování

Jeden z problémů při testování nastává v okamžiku složitějších dotazů nad větším množstvím dat v prostředí, které slouží k odzkoušení instalací potřebných programů, aplikací a daných dotazů. Toto prostředí má pouze 1GB velikost operační paměti RAM a při výpočtech složitějších dotazů aplikace neúspěšně končí s chybou OOM (Out of Memory). V produkčním prostředí, na kterém probíhá testování, už má velikost operační paměti 8 GB a veškeré testy proběhly bez problému. Velikost operační paměti 8 GB dosahují v současnosti i běžně výkonné domácí notebooky, a proto je možné tvrdit, že není potřeba k úspěšnému dokončení měření výkonností žádné výkonné databázové servery.

Další problém nastal v případě zpracovávání dotazů s prostorovými predikáty nad daty, které nejsou validní (např. data, která mají vlastní průnik sebou sama). PostGIS na kontrolu těchto objektů má speciální funkci (ST_IsValid), která vrací pravdivostní hodnotu. Bohužel tato funkce neexistuje u databázového systému SQLite, ani u dostupné verze MySQL. Proto ještě před importem samotných dat z PostgreSQL probíhá smazání prvků, které nemají validní geometrii. Tímto krokem se zajistilo, že veškerá data v databázích jsou validní.

Poslední problém se naskytl při tvorbě nové prázdné tabulky v systému SQLite. Sloupec s geometrií se vytvořil stejným způsobem jako ostatní sloupce, ovšem při importu dat systém s daným sloupcem neuměl pracovat. Tento problém se vyřeší přidáním sloupce s geometrií pomocí funkce *AddGeometryColumn(název tabulky, název sloupce s geometrii, SRID, typ geometrie, počet dimenzí).*

6. VÝSLEDKY

Výsledky testů jsou rozděleny do dvou částí (kapitoly 6.1, 6.2). V první části jsou zhodnoceny časy, jak dlouho dané testy trvaly, a také jsou zde uvedené počty množin výsledků. V druhé části jsou dané testy ze všech 3 databázových systémů porovnané formou grafů. Ke každému testu je vytvořen graf zobrazující časy všech 3 databázových systémů.

6.1.Zhodnocení databází

Tab. 15 - Výsledky testů - PostgreSQL + PostGIS

PostgreSQL + PostGIS								
Číslo testu	**1**	**2**	**3**	**4**	**5**	**6**		
Čas [s]	15.45	1.32	0.18	11	58.05	0.93		
Množina výsledků	1534	337602	98	38979	972369	140611		
Číslo testu	**7**	**8**	**9**	**10**	**11**	**12**	**13**	
Čas [s]	0.16	4.31	0.96	8.5	1.33	0.02	5.69	
Množina výsledků	12	69	307406	972369	6555	2975	218898	
Číslo testu	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**21**
Čas [s]	3.01	1.03	30.2	43.85	31.76	44.72	35.33	673.98
Množina výsledků	40750	361	1809	195	2	15164	1	169904
Číslo testu	**22**	**23**	**24**	**25**				
Čas [s]	841.48	0.15	0.98	3472.95				
Množina výsledků	388	1	1	1				

Pomocí databázového systému PostgreSQL s prostorovou nadstavbou PostGIS se zpracovaly všechny testy z navržené metodiky měření výkonnosti prostorových databází. Výsledky testů PostgreSQL jsou v Tab. 15. Z první skupiny testů, která testovala obecné vlastnosti databázových systémů, je výpočetně nejsložitější test č. 5, který vkládá nové položky do předem vytvořené tabulky, přesně se jednalo o 972 369 položek, které databázový systém vkládá do databáze průměrně za 58 s. V třetí a čtvrté skupině testů už dochází k výpočetně složitým operacím, které trvají časově delší dobu, výpočetně

nejsložitějším dotazem je test č. 25 (Vypočti míru ekologické stability krajiny v okrese Worms), který trvá v průměru 57 min a 53, 11 s. Naopak nejrychlejším dotazem je test č. 12 (Vypočti centroid transportních oblastí), tento test v průměru trvá 2 setiny sekundy.

Tab. 16 - Výsledky testů – SQLite + Spatialite

SQLite + SpatiaLite								
Číslo testu	**1**	**2**	**3**	**4**	**5**	**6**		
Čas [s]	20.4	0.8	0.1	1.73	28.94	1.75		
Množina výsledků	1534	337602	98	38979	972369	140611		
Číslo testu	**7**	**8**	**9**	**10**	**11**	**12**	**13**	
Čas [s]	0.02	5.27	1.45	13.83	1.64	0.04	2.92	
Množina výsledků	12	69	307406	972369	6555	2975	218898	
Číslo testu	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**21**
Čas [s]	4.28	91.95	7967.1	29272.3	5443.9	10723	31.94	60886
Množina výsledků	40750	361	1809	195	2	15164	1	169904
Číslo testu	**22**	**23**	**24**	**25**				
Čas [s]	8433.6	0.24	58.44	199121.95				
Množina výsledků	388	1	1	1				

Výsledky testů databázového systému SQLite s prostorovým rozšířením SpatiaLite jsou zobrazené v Tab. 16. Stejně jako u výsledků databázového systému PostgreSQL, i zde největší dobu trvá test č. 25. SpatiaLite tento dotaz zpracovává 55 hod, 18 min a 41 s. Tato doba zpracovávání je pro běžné užití nepoužitelná a je patrné, že pro tyto typy operací se databáze nehodí. Celkově za slabinu se může u tohoto databázového systému považovat zpracování prostorového spojení dvou vrstev a veškeré predikátové operace nad těmito vrstvami. Na druhou stranu, velice povzbuzujících výsledků databázový systém dosahuje při testech vyhledávání pomocí určité obálky (test č. 7), tento test v průměru trvá 2 setiny sekundy.

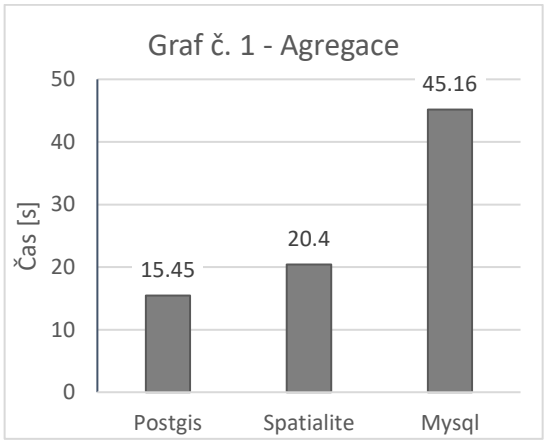
Tab. 17 - Výsledky testů - MySQL

MySQL								
Číslo testu	**1**	**2**	**3**	**4**	**5**	**6**		
Čas [s]	45.16	3.76	0.59	1.65	69.33	9693.5		
Množina výsledků	1534	337602	98	38979	972369	140611		
Číslo testu	**7**	**8**	**9**	**10**	**11**	**12**	**13**	
Čas [s]	0.03	6.07	3.11	26.67	3.1	0.06	4.55	
Množina výsledků	12	69	307406	972369	6555	2975	218898	
Číslo testu	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**21**
Čas [s]	31.99	1.85	48.25	120.25	80.53	381.58	12.44	438.88
Množina výsledků	40750	361	1809	195	2	15164	1	169904
Číslo testu	**22**	**23**	**24**	**25**				
Čas [s]	2024.1	0.24	2.16	94128.96				
Množina výsledků	388	1	1	1				

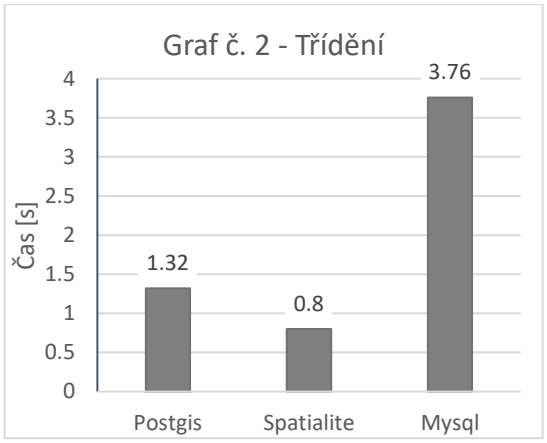
Posledním testovaným databázovým systémem je MySQL se svou prostorovou nadstavbou. Výsledky tohoto databázového systému jsou v Tab. 17. Test č. 25 tento systém zpracovává průměrně 26 hod, 8 min a 48,96 s. Tato doba je oproti zpracování PostgreSQL systémem více než 27x delší a takto náročné operace není vhodné dělat tímto databázovým systémem. Oproti ostatním databázovým systémům MySQL nedosahovala takového výkonu a vzhledem k její optimalizaci na výkon se očekávaly daleko uspokojivější výsledky, převážně u dotazů první skupiny testů. Tento fakt může být způsobený výchozím nastavením systému výrobce, což by celkově dost zhoršilo dojem na dosud velice příjemnou uživatelskou práci se systémem. MySQL dosáhlo nejlepšího výsledku při testu č. 7, daný test systém zpracovává průměrně za 3 setiny sekundy.

6.2. Porovnání databází

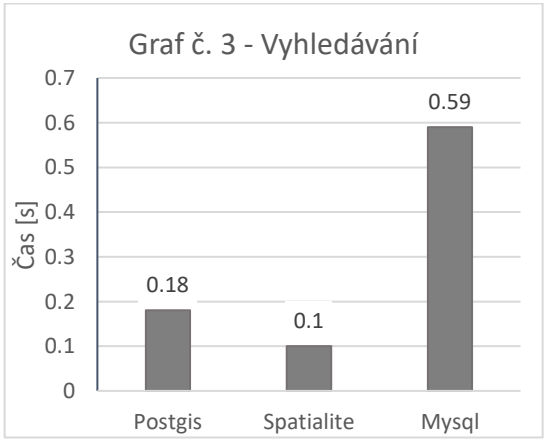
Graf 1 - Agregace



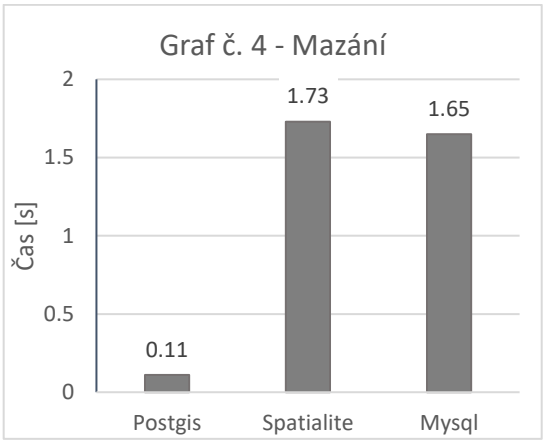
Graf 2 - Třídění



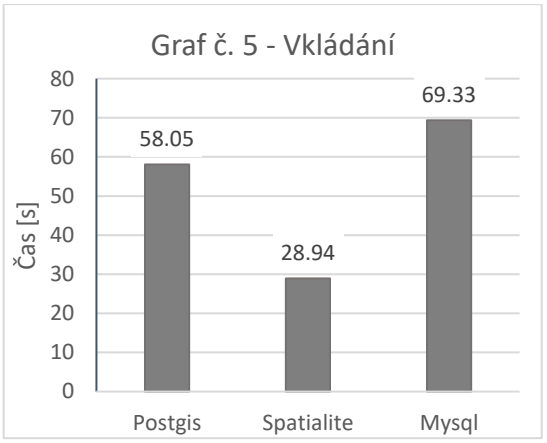
Graf 3 - Vyhledávání



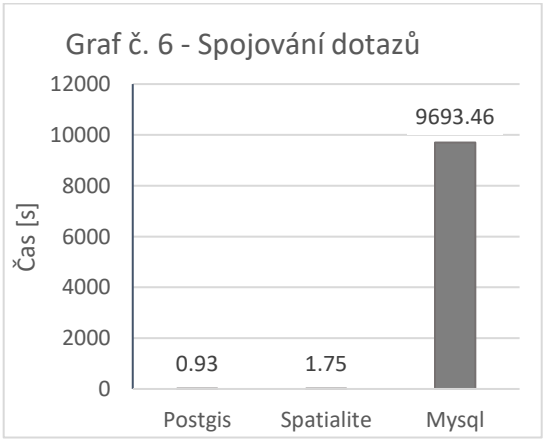
Graf 4 - Mazání



Graf 5 - Vkládání



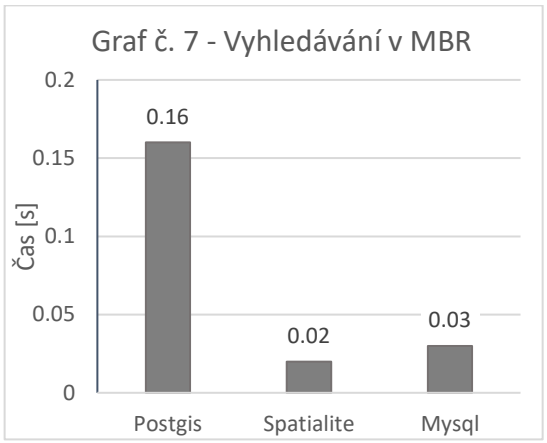
Graf 6 - Spojování dotazů



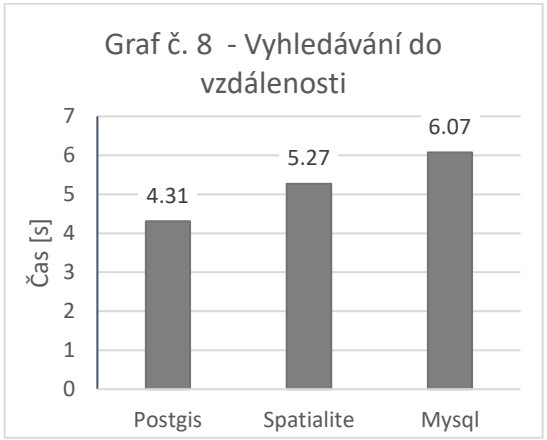
Při porovnávání databázových systémů první skupiny testů nelze s jistotou určit, který databázový systém obecně dosahuje nejlepších výsledků v první skupině testů. V testech, při

kterých dochází k agregaci, mazání nebo přirozenému spojování tabulek dosahuje nejlepších výsledků PostgreSQL, při vyhledávání nebo třídění naopak nejlepších výsledků dosahuje SQLite databázový systém. Velice rozdílné hodnoty dosahuje MySQL při dotazu nad více tabulkami s použitím přirozeného spojení INNER JOIN, zpracování testu trvá mnohonásobně déle. Tento problém nenastává při spojování tabulek přes sloupce, které obsahují vyhledávací klíč. Z tohoto případu lze tvrdit, že MySQL je oproti PostgreSQL nebo SQLite silně závislé na vyhledávacím klíči.

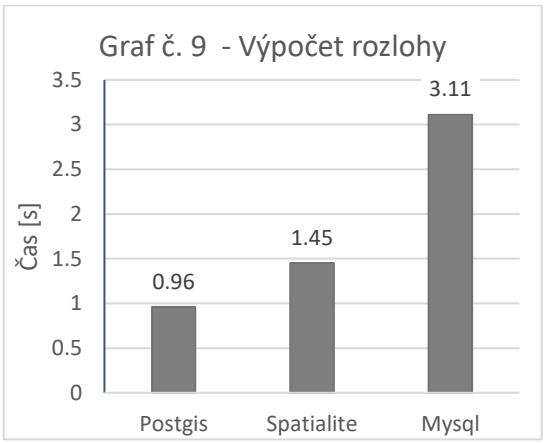
Graf 7 - Vyhledávání v MBR



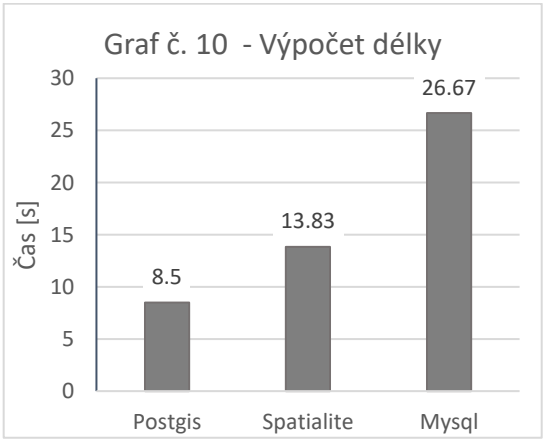
Graf 8 - Vyhledávání do vzdálenosti



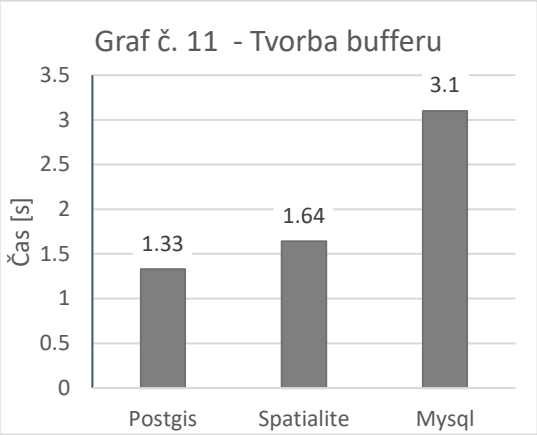
Graf 9 - Výpočet rozlohy



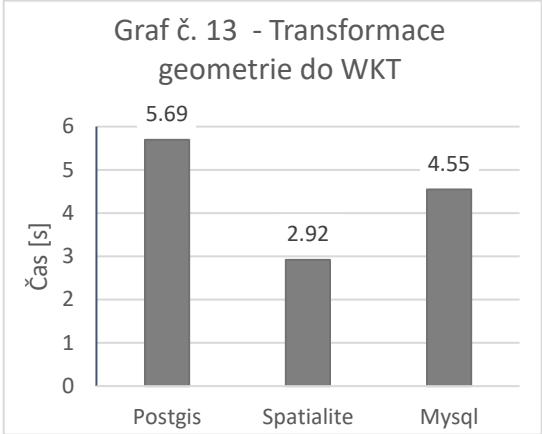
Graf 10 - Výpočet délky



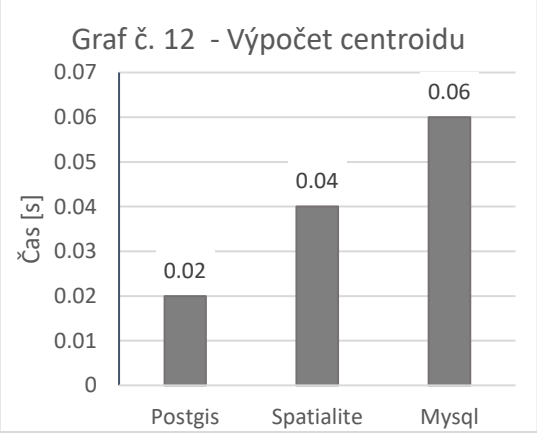
Graf 11 - Tvorba bufferu



Graf 13 - Transformace geomerie do WKT

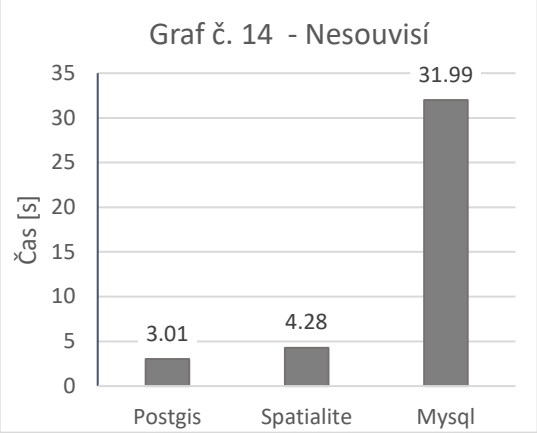


Graf 12 - Výpočet centroidu

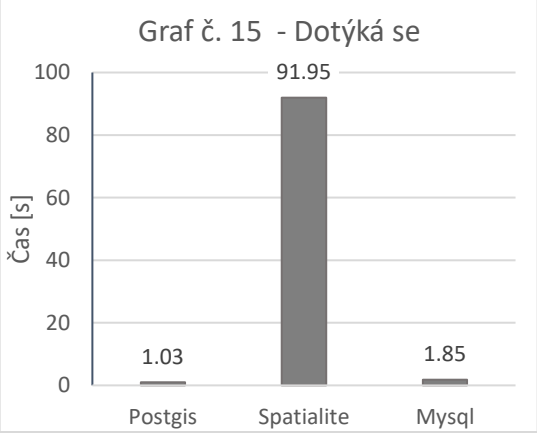


Při hodnocení druhé skupiny testů, databázový systém PostgreSQL dosahuje nejlepších výsledku, až na dotazy, při kterých se testovala transformace geometrie do formátu WKT a dotazy geometrického vyhledávání pomocí MBR. V těchto případech nejlépe zpracovával dotazy SQLite. Délky jednotlivých dotazů se v této skupině oproti ostatním dotazům liší jen v jednotkách násobků.

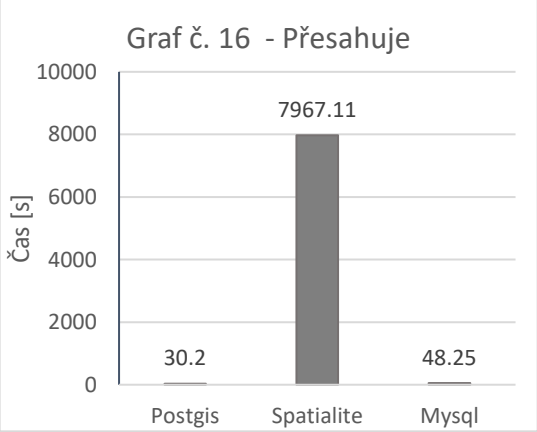
Graf 14 - Nesouvisí



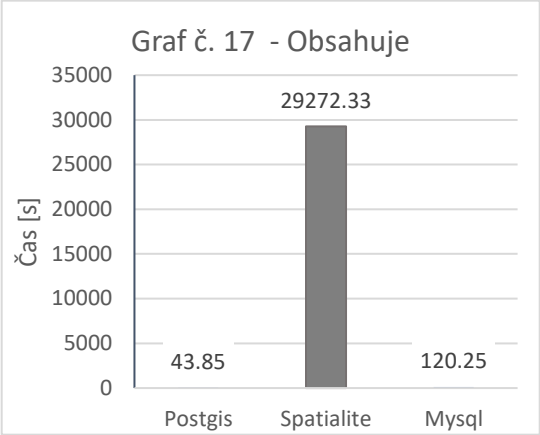
Graf 15 – Dotýká se



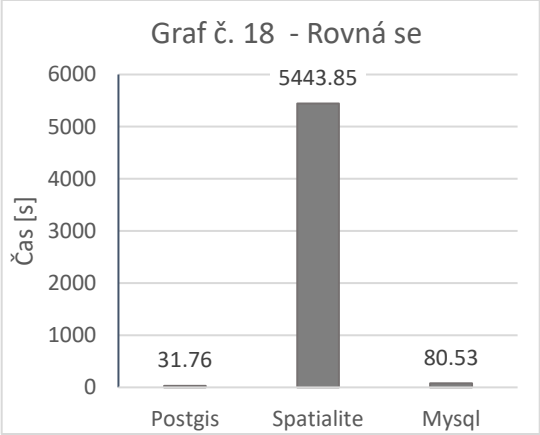
Graf 16 – Přesahuje



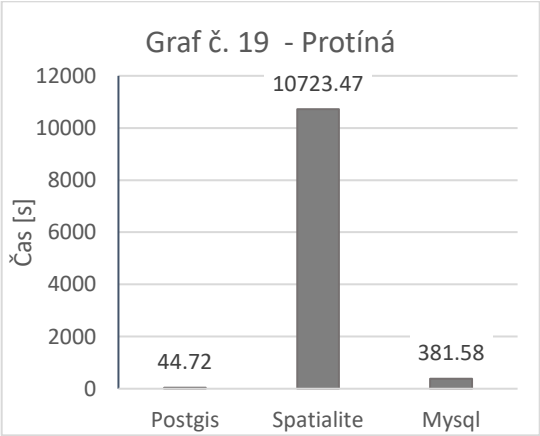
Graf 17 - Obsahuje



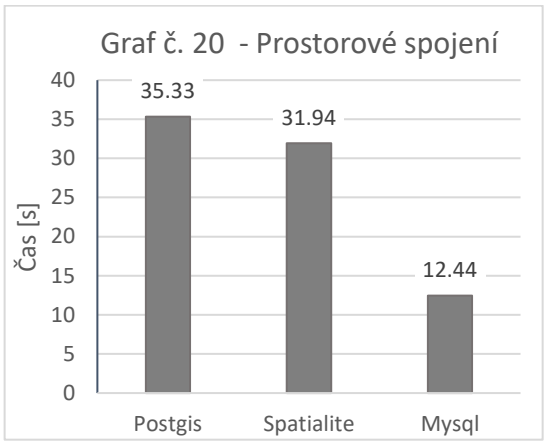
Graf 18 – Rovná se



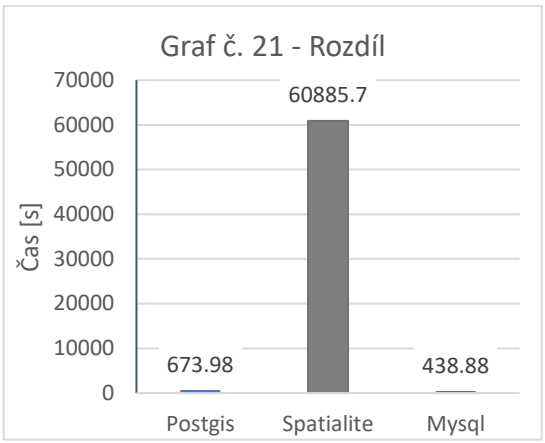
Graf 19 - Protíná



Graf 20 – Prostorové spojení

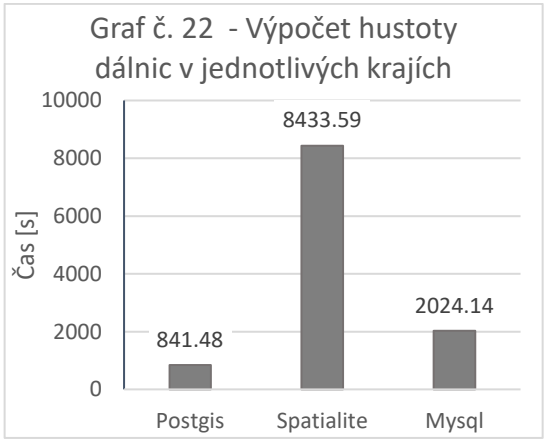


Graf 21 - Rozdíl

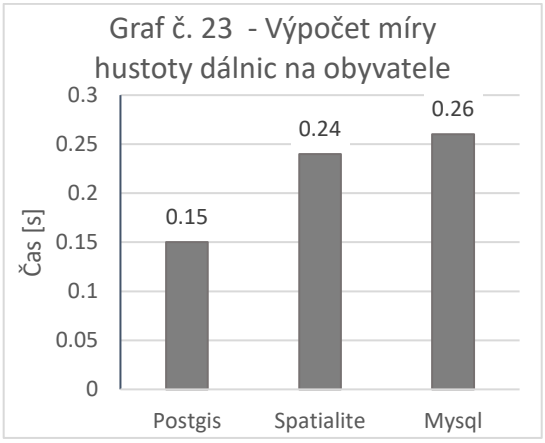


Výsledky třetí skupiny testů jsou typické tím, že ve většině těchto testů jeden databázový systém zpracovává tyto dotazy v desítkách násobků hůř než ostatní databázové systémy. Výjimkou je prostorové spojení (test č. 20), kde MySQL dosahuje nejlepšího výsledku (12,44 s), zatímco PostgreSQL dosahuje stejné množiny výsledků s průměrným časovým rozdílem téměř 21 s. Z výsledků je jasně patrné, že poměrně malá knihovna SpatiaLite výkonnostně nestačí na složité operace predikatových vztahů dvou vrstev. Nejlépe si při dotazech testující predikátové operace vedl PostgreSQL s prostorovou nadstavbou PostGIS.

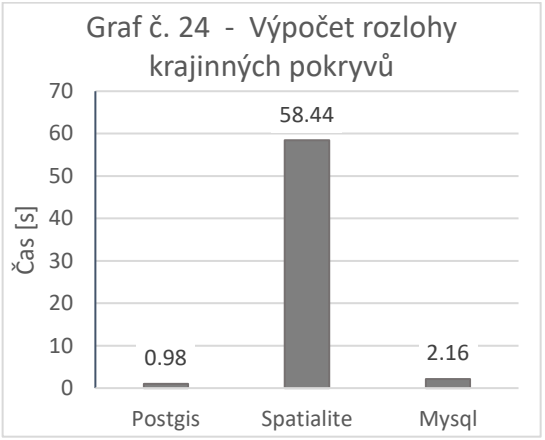
Graf 22 - Výpočet hustoty dálnic v jednotlivých krajích



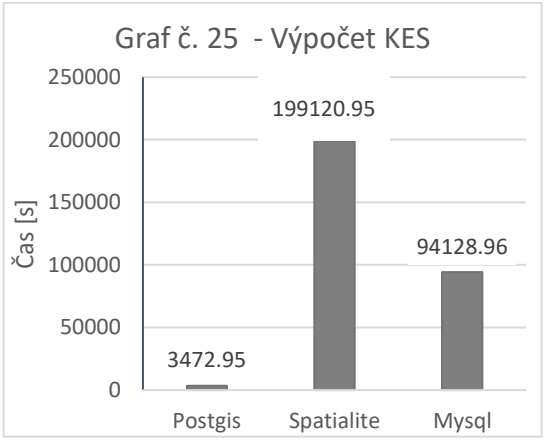
Graf 23 - Výpočet míry hustoty dálnic na obyvatele



Graf 24 - Výpočet rozlohy krajinných pokryvů



Graf 25 - Výpočet KES



Poslední skupina testů reprezentuje geoinformační operace, které se běžně vypočítávají v aplikační vrstvě systémů. Nejvíce náročným výpočtem je výpočet Ekologické stability krajiny v okrese Worms. Nejlépe si s tímto dotazem ví rady databázový systém PostgreSQL, který daný dotaz průměrně zpracovává za 3 472,95 s (57 min a 52,95 s), druhého nejlepšího výsledku dosahuje MySQL s průměrným časem 94 128,96 s (26 hod, 8 min a 48,96 s) a nejhůře si vede systém SQLite s časem 199 120,95 s (55 hod, 18 min a 40,95 s). V testech, při kterých se používají prostorové predikáty, nejhorších časů dosahuje SQLite, stejně jako v předešlé skupině testů.

Z celkového hlediska nejlepších výsledků dosahuje PostgreSQL s prostorovou nadstavbou PostGIS, který ve většině testů dominuje. SQLite dosahuje velice dobrých výsledků u jednodušších operací a pro aplikace, které nepotřebují zpracovávat složité dotazy, ale například potřebují jenom vyhledávat určité body zájmu (POI – points of interest), je tato malá knihovna dostačující. MySQL ve většině testů ztrácela oproti ostatním systémům, tento problém může způsobovat nedostatečná výchozí konfigurace, která je při instalaci systému nastavena vývojáři.

7. ZÁVĚR

Testování výkonnosti databází je velice důležitou součástí výběru vhodného databázového systému pro dobře fungující aplikaci. V oblasti tabulkových dat existuje několik testů měření výkonnosti databázových systémů, ovšem testů, které se zabývají prostorovými daty, je minimální množství a jejich stáří neumožňuje testovat nové databázové technologie.

Tato práce se zabývala tvorbou nové metodiky pro porovnávání databázových systémů, které umí pracovat s prostorovou složkou dat. Před samotným návrhem metodiky bylo potřeba se seznámit s teoretickými znalostmi databázových technologií a znalostmi testů, které se využívají pro potřeby testování databázových systémů.

Navržená metodika je založena nad datovým zdrojem OpenStreetMap z území Německa ze dne 1.1.2017 a je aplikovatelná nejenom na relační databáze, ale také na databáze nových technologií (NoSQL, Objektové databáze). Všechny prostorové operace, které jsou v metodice zařazené, podléhají standardizaci organizace OGC a samotný testovací scénář metodiky netestuje pouze schopnosti práce s prostorovými daty, ale také obecné vlastnosti databázových systémů. Cílem metodiky je umožnit všem zájemcům zobrazení všech výsledků, konfigurací, dotazů a parametrů stroje, na kterém docházelo k testování. Metodika je rozdělena do 4 kategorií podle účelů testů (kapitola 4).

Výběr dat z projektu OpenStreetMaps mělo pro testování několik výhod, datový zdroj byl dostatečně kvalitní a rozsáhlý pro široce využitelné testování prostorových databází. Dalšími výhodami je licencování těchto dat, díky kterým má přístup k datům kdokoliv bez omezení i pro komerční účely. Výhodou použití OSM pro tvorbu testování prostorových databází je volná dostupnost datových zdrojů určitých oblastí a z určitých časových období. Popularita práce s těmito daty ve sféře open source a free software zapříčinila široké množství různých nástrojů pro import OSM dat do různých databázových řešení a proto není import do databází složitý.

Pro ověření navržené metodiky byly vybrány 3 databázové systémy – PostgreSQL

s prostorovou nadstavbou PostGIS, SQLite s prostorovým rozšířením SpatiaLite a MySQL se svým prostorovým rozšířením. Na těchto databázích byly provedené testy a výsledky jsou zobrazené v tabulkách (Tab. 15, Tab. 16, Tab. 17), grafech (Graf 1 – Graf 25) a také jsou verbálně zhodnocené. Výsledky ukazují, že databáze PostgreSQL dosahuje velice dobrých výsledků, a i při výchozím nastavení systému na slabém testovaném stroji umožňuje plnohodnotné využití v GIS. SQLite s prostorovým rozšířením SpatiaLite dosahovalo uspokojivých výsledků při relativně jednoduchých výpočtech, v těchto výpočtech mnohdy překonával PostgreSQL. Tento databázový systém může být určitě vhodný pro jednoduchou aplikaci, při které dochází pouze k dotazům bez prostorových predikátů, ve kterých měl tento databázový systém největší slabinu. Posledním systémem byl MySQL, jehož výsledky při výchozí konfiguraci systému byly výkonnostně slabší oproti konkurentům. Správné nastavení konfigurace by ovšem bylo nad rámec této práce. Samotné výsledky jsou zveřejněné s konfiguracemi jednotlivých databázových systémů, s parametry stroje, na kterém probíhalo testování a s dotazy, které se použily k získání úspěšné množiny výsledků. Konfigurace systémů a dotazy jazyka SQL jednotlivých databázových systémů jsou v přílohách (Příloha 1 – Příloha 5).

Metodika vymyšlená v této práci byla úspěšně reprezentovaná na 3 různých databázových systémech. Na relativně slabém stroji nenastaly během testů žádné vážné problémy. Výsledky testů metodiky odhalily určité problémy, které jednotlivé databázové systémy mají a všechny výsledky testů, které byly reprezentovány na 3 různých databázových systémech, jsou od sebe jednoznačně odlišitelné. Z výsledných množin prvků daných dotazů, které pro všechny 3 databázové systémy vyšly totožně, lze usuzovat, že metodický vzorek dat (OSM na území Německa z 1.1.2017) neobsahuje žádné nejasné situace a je možné ho věrohodně použít pro porovnávání dalších databázových systémů.

ZDROJE

Literatura

Adminer. 2016. Adminer – Správa databáze v jednom PHP souboru. [online]. [cit. 2016-10-17]. Dostupné z WWW: <https://www.adminer.org/cs/>

BLUM, Richard. 2007. PostgreSQL 8 for windows. New York: McGraw-Hill. ISBN 0071485627.

BŘEHOVSKÝ, Martin & JEDLIČKA, K. 2016. Úvod do geografických informačních systémů. Přednáškový text. [online]. [cit. 2016-10-17]. Dostupné z WWW: <http://gis.zcu.cz/studium/ugi/e-skripta/ugi.pdf>

CATTELL, R.G.G. 1994. The Object Database Standard: Odmg-93. San Francisco., CA. Morgan Kaufmann Publishers. 184 s. ISBN 978-1558603967

CODD, E.F. 1970. A Relational Model of Data for Large Shared Data Banks. IBM Research Lab, San Jose, CA.

CRHONEK, Tomáš. 2011. Optimalizace databáze PostgreSQL. [online]. Linux Expres. [cit. 2016-10-17]. Dostupné z: <https://www.linuxexpres.cz/praxe/optimalizace-databaze-postgresql>

Geofabrik. 2016. OpenStreetMap Data Extracts. [online]. Geofabrik GmbH [cit. 2016-10-17]. Dostupné z WWW: <http://download.geofabrik.de/>

GILFILIAN, I. 2003. *Myslíme v MySQL 4, knihovna programátora.* Praha. Grada Publishing, a.s. 750 s. ISBN80-247-0661-X.

HeidiSQL. 2017. HeidiSQL. [online]. [cit. 2016-10-17]. Dostupné z WWW: <https://www.heidisql.com/>

HUPPLER, Karl. 2009.The Art of Building Good Benchmark. First TPC Technology

Conference, TPCTC 2009, Revised Selected Papers. Berlin.

Internet CZ. 2017. *Kompletní cloudové prostředí pro vývoj vašich projektů*. [online]. [cit. 2016-10-19]. Dostupné z WWW: <https://www.forpsicloud.cz/ >

JEDLINSKÝ, Jan. 2006. *Způsoby uložení prostorových dat v databázi pro účely pozemkového datového modelu.* Plzeň. 56 s. Diplomová práce. Západočeský univerzita. Ved. práce Karel Jedlička.

LACKO, Ľuboslav. 2011. *1001 tipů a triků pro SQL*. Vydání první. Brno. Computer Press, a.s. 258 s., ISBN 978-80-251-3010-0

LANDA, Martin. 2016. *DE-9IM, funkce prostorové analýzy. Studijní materiál k předmětu* Úvod do zpracování prostorových dat. ČVUT. [online]. [cit. 2016-10-19]. Dostupné z WWW: <http://geo.fsv.cvut.cz/~gin/uzpd/uzpd-02-de-9im.pdf>

KANDASAMY, Saravanakumar. 2015. *How to choose indexing or hashing technique?* [online]. [cit. 2016-10-17]. Dostupné z WWW: <http://www.exploredatabase.com/search/label/Indexing?updated-max=2016-02-17T22:06:00%2B05:30&max-results=20&start=4&by-date=false>

KOLÁŘ, Dušan. 2006. *Pokročile databázové systémy (Prostorové databáze).* Brno. Studijní opora. FIT VUT v Brně.

KOŠÁREK, Lukáš. 2010. *Výkonnostní srovnání relačních databází*. Brno. 41 s., Bakalářská práce. Masarykova univerzita. Ved. práce RNDr. Vlastislav Dohnal, Ph.D.

MERTZ, David. 2001. *Putting XML in context with hierarchical, relational and object-oriented models*. [online]. [cit. 2017-1-28]. Dostupné z WWW: <https://www.ibm.com/developerworks/xml/library/x-matters8/index.html>

MySQL Engines. 2017. *MySQL – Alternative Storage Engines*. [online]. [cit. 2017-1-16]. Dostupné z WWW: <https://dev.mysql.com/doc/refman/5.7/en/storage-engines.html>

MySQL Spatial. 2017. *MySQL – Extension for Spatial Data*. [online]. [cit. 2017-1-16]. Dostupné z WWW: <https://dev.mysql.com/doc/refman/5.7/en/spatial-extensions.html>

NEUBAUER, Peter. 1999. *B-Trees: Balanced Tree Data Structures.* [online]. [cit. 2016-10-23]. Dostupný z WWW: <http://www.bluerwhite.org/btree/>

OGC about. 2016. *About OGC*. [online]. [cit. 2016-11-07]. Dostupné z WWW: <http://www.opengeospatial.org/ogc>

OGC standards. 2016. *OGC Standards and Supporting Documents*. [online]. [cit. 2016-11-07]. Dostupné z WWW: <http://www.opengeospatial.org/standards>

OLSZOWSKI, Pavel & FARANA, Radim. 1997. *Dotazovací jazyk SQL.* Ostrava. [online]. [cit. 2017-1-23]. Dostupný z WWW: <http://books.fs.vsb.cz/SQLReference/>

OpenStreetMap. 2017. *OpenStreetMap Foundation.* [online]. OpenStreetMap Foundation [cit. 2017-1-23]. Dostupný z WWW: <http://wiki.osmfoundation.org/wiki/Main_Page>

OpenStreetMap – Copyright. 2017. *OpenStreetMap – Copyright and License.* [online]. OpenStreetMap Foundation. [cit. 2017-1-23]. Dostupný z WWW: <https://www.openstreetmap.org/copyright>

PAVELKA, K. aj. 2005. *Terminilogický slovník zeměměřictví a katastru nemovitostí.* [online]. [cit. 2017-1-23]. Dostupný z WWW: <http://www.vugtk.cz/slovnik/index.php>

PAZDERA, Miroslav. 2007. *Porovnávací studie databází*. Brno. 51 s., Bakalářská práce. Vysoké učení technické v Brně. Ved. práce Doc. Ing. Branislav Lacko, CSc.

PgAdmin. 2017. *PgAdmin PostgreSQL Tools*. [online]. [cit. 2016-10-12]. Dostupný z WWW: <https://www.pgadmin.org/>

PhpMyAdmin. 2017. *PhpMyAdmin – Official web*. [online]. [cit. 2016-10-12]. Dostupný z

WWW: <https://www.phpmyadmin.net/>

PhpPgAdmin. 2013. *PhpPgAdmin – web-based administration tool.* [online]. [cit. 2017-10-12]. Dostupný z WWW: <http://phppgadmin.sourceforge.net/doku.php>

POKORNÝ, Jaroslav. 2000. *Prostorové datové struktury a jejich použití k indexaci prostorových objektů*. Praha. MFF UK. [online]. [cit. 2016-10-12]. Dostupný z WWW: <http://gisak.vsb.cz/GIS_Ostrava/GIS_Ova_2000/Sbornik/Pokorny/Referat.htm>

POKORNÝ, Jaroslav & VALENTA, Michal. 2005. *Objektově relační database*. Praha. ČVUT. [online]. [cit. 2016-10-12]. Dostupný z WWW: <https://users.fit.cvut.cz/valenta/doku/lib/exe/fetch.php/bivs/dbs2_05_ordbms.pdf>

PostGIS reference. 2016. *PostGIS Reference*. [online]. PostgreSQL Global Development Group. [cit. 2016-10-12]. Dostupný z WWW: <http://postgis.net/docs/reference.html>

PostgreSQL about. 2016. *PostgreSQL*. [online]. PostgreSQL Global Development Group. [cit. 2016-10-12]. Dostupný z WWW: <https://www.postgresql.org/about/>

PostgreSQL documentation. 2017. *PostgreSQL 9.5.6 Documentation*. [online]. PostgreSQL Global Development Group. [cit. 2016-1-12]. Dostupný z WWW: <https://www.postgresql.org/docs/9.5/static/index.html>

PRABHU, C.S.R. 1998. *Object-oriented database systems: approaches and architectures.* New Delhi. Prentice-Hall of India. ISBN 8120312570.

PSQL. 2017. *Psql – PostgreSQL interactive terminal*. [online]. [cit. 2017-10-12]. Dostupný z WWW: <https://www.postgresql.org/docs/9.5/static/app-psql.html>

REID, G. Elizabeth. 2009. *Design and Evaluation of a Benchmark for Main Memory Transaction Processing Systems*. Massachusetts. 63 s., Diplomová práce. Massachusetts institute of technology. Ved. práce Samuel Madden.

RYDVAL, Slávek. 2005. *Historie jazyka SQL*. [online]. [cit. 2017-1-27]. Dostupný z WWW: <http://www.rydval.cz/phprs/view.php?cisloclanku=2005123125>

SHAMKANT, B. Navathe. 2003. *Fundamentals of Database* Systems. 5. vydání. Boston. Pearson/Addison Wesley. 1029 s. ISBN 0-321-12226-7

SHANLEY, Kim. 1998. *History and Overview of the TPC*. Transaction Processing Performance Council. [online]. [cit. 12.12.2016]. Dostupný z WWW: <http://www.tpc.org/information/about/history.asp>

SHASHANK, Tiwari. 2011. *Professional NoSQL.* Hoboken, N.J. Wiley. 384 s. ISBN9780470942246.

SILBERSCHATZ aj. 2006. SILBERSCHATZ, A., KORTH, H. F., SUDARSHAN, S. *Database System Concepts*. 5. vyd. New York: McGraw-Hill. 2006. 1142 s. ISBN978-0-07-295886-7

SLÍŽEK, David. 2014. *Mapy.cz vyměnily podklady pro Evropu, používají OpenStreetMap*. [online]. Lupa.cz. [cit. 10.12.2016]. Dostupný z WWW: <http://www.lupa.cz/clanky/mapy-cz-vymenily-podklady-pro-evropu-pouzivaji-openstreetmap/>

SpatiaLite. 2013. *SpatiaLite 2.3.1 – Architecture*. [online]. [cit. 10.12.2016]. Dostupný z WWW: <http://www.gaia-gis.it/spatialite-2.3.1/spatialite-arch-2.3.1.html>

SpatiaLite-GIS. 2017. *SpatiaLite – GIS*. [online]. [cit. 27.11.2017]. Dostupný z WWW: <https://www.gaia-gis.it/fossil/spatialite_gis/index>

SpatiaLite-GIS Advanced. 2015. *Not at all a manual simple a quick how-to-do guide.* [online]. [cit. 15.2.2017]. Dostupný z WWW: <http://www.gaia-gis.it/gaia-sins/spatialite-gis-docs/spatialite-gis-advanced.pdf>

SpatiaLite-GIS Basic. 2015. *Not at all a manual simple a quick how-to-do guide.* [online]. [cit. 15.2.2017]. Dostupný z WWW: <http://www.gaia-gis.it/gaia-sins/spatialite-gis-

docs/spatialite-gis-basic.pdf>

SPEC. 2009. *Standard Performance Evaluation Corporation*. [online]. [cit. 16.10.2016]. The SPEC Organization. Dostupné z WWW: <http://www.spec.org/spec/index.html>.

SQLite About. 2016. *About SQLite.* SQLite Consortium. [online]. [cit. 16.10.2016]. Dostupný z WWW: <https://sqlite.org/about.html>

SQLite browser. 2016. *DB Browser for SQLite.* SQLite Consortium. [online]. [cit. 27.11.2017]. Dostupný z WWW: <http://sqlitebrowser.org/>

SQLite difference. 2016. *Distinctive Features Of SQLite*. SQLite Consortium. [online]. [cit. 27.11.2017]. Dostupný z WWW: <http://www.sqlite.org/different.html>

SQLite Documentation. 2016. *SQLite Documentation*. SQLite Consortium. [online]. [cit. 16.10.2016]. Dostupný z WWW: <https://sqlite.org/docs.html>

STONEBRAKER aj. 1993. *The Sequoia 2000 Storage Benchmark*. 14 s., University of California, Berkeley.

STĚHULE, P. 2012. Historie projektu PostgreSQL. [online]. Root.cz [cit. 27.11.2017]. Dostupný z WWW: <https://www.root.cz/clanky/historie-projektu-postgresql/>

TPC-C. 1997. *TPC-C is an on-line transaction processing benchmark*. Transaction Processing Performance Council. [online]. [cit. 12.12.2016]. Dostupný z WWW: <http://www.tpc.org/tpcc/default.asp>

TRODD, N. 2012. *Spatial Data Models and Spatial Data Structures*. [online]. [cit. 2017-1-25]. Dostupné z WWW: <http://www.gisknowledge.net/topic/methods_in_gis/trodd_spatial_data_models_and_spatial_data_structures_05.pdf>

VALENTA, Michal. 2016. *Fyzické uložení dat a indexy*. Studijní materiál k předmětu

Databázové systémy. Praha. ČVUT. [online]. [cit. 2017-1-27]. Dostupné z WWW: <http://sklad.izmus.cz/cvut/dbs/dbs_lec_10_fyzicke_ulozeni_dat.pdf>

VALENTA, Michal. 2011. *DBS – Databázové modely.* Studijní material k předmětu Databázové systémy. Praha. ČVUT. [online]. [cit. 2017-1-27]. Dostupné z WWW: <https://users.fit.cvut.cz/valenta/doku/lib/exe/fetch.php/bivs/dbs_02_databazove_modely.pdf>

VIRT, Jan. 2010. *Měření výkonnosti relačních databází.* Praha. 61 s., Bakalářská práce. Bankovní institut vysoká škola Praha. Ved. práce Ing. Michal Valenta, Ph.D.

VRBÍK, Tomáš. 2012. *Srovnání distribuovaných „NoSQL" databází s důrazem na výkon a škálovatelnost*. Praha. 82 s., Diplomová práce. Vysoká škola ekonomická v Praze. Ved. práce Mgr. Zbyněk Šlajchrt.

WELLING, Luke & THOMSON, Laura. 2005. *MySQL Průvodce základy databázového systému.* Brno. CP Books, a. s. 2005. 250 s. ISBN80-251-0671-3.

WERSTEIN, Paul. 1998. *A performance Benchmark for Spatiotemporal Databases.* Dunedin. 9 s., University of Otago.

Wiki OSM. 2014. *Wiki OpenStreetMap*. [online]. [cit. 2017-1-27]. Dostupné z WWW: <http://wiki.openstreetmap.org/wiki/Main_Page>

WILLIS, Nathan. 2007. *OpenStreetMap project imports US government maps*. [online]. Linux.com [cit. 2017-1-27]. Dostupné z WWW: <https://www.linux.com/news/openstreetmap-project-imports-us-government-maps?theme=print>

WON, Kim. 1995. *Modern Database Systems*. ACM Press, 1995, ISBN 0-201-59098-0

Použité tutoriály k software

GDAL. 2017. GDAL – Geospatial Data Abstraction Library. [online]. [cit. 2016-10-17]. Dostupné z: <http://www.gdal.org/>

Imposm. 2016. Imports OpenStreetMap data into PostgreSQL/PostGIS databases. [online]. Omniscale GmbH & Co. KG [cit. 2016-10-19]. Dostupné z WWW: <https://imposm.org/>

JACOLIN, Y. 2016. *Osm2pgsql to imposm schema.sql.* [online]. [cit. 2016-10-19]. Dostupné z WWW: <https://github.com/mapserver/basemaps/blob/master/contrib/osm2pgsql-to-imposm-schema.sql>

KYSILKA, Pavel. 2003. Praktický návod k PgSQL. [online]. ABC Linuxu [cit. 2016-11-26]. Dostupné z WWW: <http://www.abclinuxu.cz/clanky/navody/prakticky-navod-k-pgsql>

MySQL Reference. 2017. MySQL - Reference Manual. [online]. [cit. 2017-1-16]. Dostupné z WWW: <https://dev.mysql.com/doc/refman/5.7/en/>

PostGIS manual. 2016. PostGIS Manual. [online]. PostgreSQL Global Development Group. [cit. 2016-10-12]. Dostupný z WWW: <http://postgis.net/docs/manual-2.0/>

SpatiaLite – Reference. 2015. SpatiaLite 4.0.0 SQL functions reference list. [online]. [cit. 15.2.2017]. Dostupný z WWW: <http://www.gaia-gis.it/gaia-sins/spatialite-sql-4.2.0.html>

Použitá data

Data OpenStreetMap – Německo, ze dne 1.1.2017, dostupné z (Geofabrik, 2016).

Použitý software

PostgreSQL 9.5.6
PostGIS 2.0
SQLite 3.11.0
SpatiaLite 4.3.0
MySQL 5.7.17

PŘÍLOHY

V přílohách se nachází konfigurační soubory databázových systémů, které jsou použité k testování (Příloha 1, Příloha 2). Dále jsou součástí příloh databázové dotazy jednotlivých databází (Příloha 3, Příloha 4 a Příloha 5).

Příloha 1 - Konfigurační soubor databázového systému PostgreSQL

```
# -----------------------------
# PostgreSQL configuration file
# -----------------------------
## This file consists of lines of the form:
#       name = value
## (The "=" is optional.)  Whitespace may be used.  Comments are introduced with
# "#" anywhere on a line.  The complete list of parameter names and allowed
# values can be found in the PostgreSQL documentation.
## The commented-out settings shown in this file represent the default values.
# Re-commenting a setting is NOT sufficient to revert it to the default value;
# you need to reload the server.
## This file is read on server startup and when the server receives a SIGHUP
# signal.  If you edit the file on a running system, you have to SIGHUP the
# server for the changes to take effect, or use "pg_ctl reload".  Some
# parameters, which are marked below, require a server shutdown and restart to
# take effect.
```

```
## Any parameter can also be given as a command-line option to the server, e.g.,
# "postgres -c log_connections=on".  Some parameters can be changed at run time
# with the "SET" SQL command.
## Memory units:      kB = kilobytes        Time units:    ms  = milliseconds
#                     MB = megabytes                       s = seconds
#                     GB = gigabytes                       min = minutes
#                     TB = terabytes                       h   = hours
#                                                          d   = days
#------------------------------------------------------------------------------
# FILE LOCATIONS
#------------------------------------------------------------------------------
# The default values of these variables are driven from the -D command-line
# option or PGDATA environment variable, represented here as ConfigDir.
data_directory = '/var/lib/postgresql/9.5/main'         # use data in another directory
                                        # (change requires restart)
hba_file = '/etc/postgresql/9.5/main/pg_hba.conf'    # host-based authentication file
                                        # (change requires restart)
ident_file = '/etc/postgresql/9.5/main/pg_ident.conf'          # ident configuration file
                                        # (change requires restart)
# If external_pid_file is not explicitly set, no extra PID file is written.
external_pid_file = '/var/run/postgresql/9.5-main.pid'                  # write an extra
PID file
                                        # (change requires restart)
#------------------------------------------------------------------------------
# CONNECTIONS AND AUTHENTICATION
#------------------------------------------------------------------------------
# - Connection Settings -
#listen_addresses = 'localhost'                 # what IP address(es) to listen on;
                                        # comma-separated list of addresses;
                                        # defaults to 'localhost'; use '*' for all
                                        # (change requires restart)
port = 5432                             # (change requires restart)
max_connections = 100                       # (change requires restart)
```

```
#superuser_reserved_connections = 3         # (change requires restart)
unix_socket_directories = '/var/run/postgresql'       # comma-separated list of directories
                                        # (change requires restart)
#unix_socket_group = ''                     # (change requires restart)
#unix_socket_permissions = 0777             # begin with 0 to use octal notation
                                        # (change requires restart)
#bonjour = off                      # advertise server via Bonjour
                                        # (change requires restart)
#bonjour_name = ''                  # defaults to the computer name
                                        # (change requires restart)
# - Security and Authentication -
#authentication_timeout = 1min              # 1s-600s
ssl = true                              # (change requires restart)
#ssl_ciphers = 'HIGH:MEDIUM:+3DES:!aNULL' # allowed SSL ciphers
                                        # (change requires restart)
#ssl_prefer_server_ciphers = on             # (change requires restart)
#ssl_ecdh_curve = 'prime256v1'              # (change requires restart)
ssl_cert_file = '/etc/ssl/certs/ssl-cert-snakeoil.pem'          # (change requires restart)
ssl_key_file = '/etc/ssl/private/ssl-cert-snakeoil.key'         # (change requires restart)
#ssl_ca_file = ''                   # (change requires restart)
#ssl_crl_file = ''                  # (change requires restart)
#password_encryption = on
#db_user_namespace = off
#row_security = on
# GSSAPI using Kerberos
#krb_server_keyfile = ''
#krb_caseins_users = off
# - TCP Keepalives -
# see "man 7 tcp" for details
#tcp_keepalives_idle = 0            # TCP_KEEPIDLE, in seconds;
                                        # 0 selects the system default
#tcp_keepalives_interval = 0        # TCP_KEEPINTVL, in seconds;
                                        # 0 selects the system default
```

```
#tcp_keepalives_count = 0          # TCP_KEEPCNT;
                                   # 0 selects the system default
#------------------------------------------------------------------------------
# RESOURCE USAGE (except WAL)
#------------------------------------------------------------------------------
# - Memory -
shared_buffers = 128MB                  # min 128kB
                                   # (change requires restart)
#huge_pages = try                  # on, off, or try
                                   # (change requires restart)
#temp_buffers = 8MB                     # min 800kB
#max_prepared_transactions = 0          # zero disables the feature
                                   # (change requires restart)
# Caution: it is not advisable to set max_prepared_transactions nonzero unless
# you actively intend to use prepared transactions.
#work_mem = 4MB                         # min 64kB
#maintenance_work_mem = 64MB            # min 1MB
#autovacuum_work_mem = -1               # min 1MB, or -1 to use
maintenance_work_mem
#max_stack_depth = 2MB                  # min 100kB
dynamic_shared_memory_type = posix      # the default is the first option
                                   # supported by the operating system:
                                   #   posix
                                   #   sysv
                                   #   windows
                                   #   mmap
                                   # use none to disable dynamic shared memory
# - Disk -
#temp_file_limit = -1               # limits per-session temp file space
                                   # in kB, or -1 for no limit
# - Kernel Resource Usage -
#max_files_per_process = 1000           # min 25
                                   # (change requires restart)
```

```
#shared_preload_libraries = ''          # (change requires restart)
# - Cost-Based Vacuum Delay -
#vacuum_cost_delay = 0                  # 0-100 milliseconds
#vacuum_cost_page_hit = 1               # 0-10000 credits
#vacuum_cost_page_miss = 10             # 0-10000 credits
#vacuum_cost_page_dirty = 20            # 0-10000 credits
#vacuum_cost_limit = 200                # 1-10000 credits
# - Background Writer -
#bgwriter_delay = 200ms                 # 10-10000ms between rounds
#bgwriter_lru_maxpages = 100            # 0-1000 max buffers written/round
#bgwriter_lru_multiplier = 2.0          # 0-10.0 multipler on buffers scanned/round
# - Asynchronous Behavior -
#effective_io_concurrency = 1           # 1-1000; 0 disables prefetching
#max_worker_processes = 8
#------------------------------------------------------------------------------
# WRITE AHEAD LOG
#------------------------------------------------------------------------------
# - Settings -
#wal_level = minimal                    # minimal, archive, hot_standby, or logical
                                        # (change requires restart)
#fsync = on                             # turns forced synchronization on or off
#synchronous_commit = on                # synchronization level;
                                        # off, local, remote_write, or on
#wal_sync_method = fsync                # the default is the first option
                                        # supported by the operating system:
                                        #   open_datasync
                                        #   fdatasync (default on Linux)
                                        #   fsync
                                        #   fsync_writethrough
                                        #   open_sync
#full_page_writes = on                  # recover from partial page writes
#wal_compression = off                  # enable compression of full-page writes
#wal_log_hints = off                    # also do full page writes of non-critical updates
```

```
                                        # (change requires restart)
#wal_buffers = -1                      # min 32kB, -1 sets based on shared_buffers
                                        # (change requires restart)
#wal_writer_delay = 200ms              # 1-10000 milliseconds
#commit_delay = 0                      # range 0-100000, in microseconds
#commit_siblings = 5                   # range 1-1000
# - Checkpoints -
#checkpoint_timeout = 5min             # range 30s-1h
#max_wal_size = 1GB
#min_wal_size = 80MB
#checkpoint_completion_target = 0.5         # checkpoint target duration, 0.0 - 1.0
#checkpoint_warning = 30s              # 0 disables
# - Archiving -
#archive_mode = off          # enables archiving; off, on, or always
                            # (change requires restart)
#archive_command = ''                  # command to use to archive a logfile segment
                            # placeholders: %p = path of file to archive
                            #               %f = file name only
                            # e.g. 'test ! -f /mnt/server/archivedir/%f && cp %p
/mnt/server/archivedir/%f'
#archive_timeout = 0         # force a logfile segment switch after this
                            # number of seconds; 0 disables
#------------------------------------------------------------------------------
# REPLICATION
#------------------------------------------------------------------------------
# - Sending Server(s) -
# Set these on the master and on any standby that will send replication data.
#max_wal_senders = 0                   # max number of walsender processes
                            # (change requires restart)
#wal_keep_segments = 0                 # in logfile segments, 16MB each; 0 disables
#wal_sender_timeout = 60s  # in milliseconds; 0 disables
#max_replication_slots = 0  # max number of replication slots
                            # (change requires restart)
```

```
#track_commit_timestamp = off      # collect timestamp of transaction commit
                                   # (change requires restart)
# - Master Server -
# These settings are ignored on a standby server.
#synchronous_standby_names = ''    # standby servers that provide sync rep
                                   # comma-separated list of application_name
                                   # from standby(s); '*' = all
#vacuum_defer_cleanup_age = 0      # number of xacts by which cleanup is delayed
# - Standby Servers -
# These settings are ignored on a master server.
#hot_standby = off                 # "on" allows queries during recovery
                                   # (change requires restart)
#max_standby_archive_delay = 30s # max delay before canceling queries
                                   # when reading WAL from archive;
                                   # -1 allows indefinite delay
#max_standby_streaming_delay = 30s        # max delay before canceling queries
                                   # when reading streaming WAL;
                                   # -1 allows indefinite delay
#wal_receiver_status_interval = 10s # send replies at least this often
                                   # 0 disables
#hot_standby_feedback = off        # send info from standby to prevent
                                   # query conflicts
#wal_receiver_timeout = 60s        # time that receiver waits for
                                   # communication from master
                                   # in milliseconds; 0 disables
#wal_retrieve_retry_interval = 5s  # time to wait before retrying to
                                   # retrieve WAL after a failed attempt
#------------------------------------------------------------------------------
# QUERY TUNING
#------------------------------------------------------------------------------
# - Planner Method Configuration -
#enable_bitmapscan = on
#enable_hashagg = on
```

```
#enable_hashjoin = on
#enable_indexscan = on
#enable_indexonlyscan = on
#enable_material = on
#enable_mergejoin = on
#enable_nestloop = on
#enable_seqscan = on
#enable_sort = on
#enable_tidscan = on
# - Planner Cost Constants -
#seq_page_cost = 1.0                    # measured on an arbitrary scale
#random_page_cost = 4.0                 # same scale as above
#cpu_tuple_cost = 0.01                  # same scale as above
#cpu_index_tuple_cost = 0.005           # same scale as above
#cpu_operator_cost = 0.0025             # same scale as above
#effective_cache_size = 4GB
# - Genetic Query Optimizer -
#geqo = on
#geqo_threshold = 12
#geqo_effort = 5                        # range 1-10
#geqo_pool_size = 0                     # selects default based on effort
#geqo_generations = 0                   # selects default based on effort
#geqo_selection_bias = 2.0              # range 1.5-2.0
#geqo_seed = 0.0                        # range 0.0-1.0
# - Other Planner Options -
#default_statistics_target = 100        # range 1-10000
#constraint_exclusion = partition       # on, off, or partition
#cursor_tuple_fraction = 0.1            # range 0.0-1.0
#from_collapse_limit = 8
#join_collapse_limit = 8                # 1 disables collapsing of explicit
                                        # JOIN clauses
#------------------------------------------------------------------------------
# ERROR REPORTING AND LOGGING
```

```
#------------------------------------------------------------------------------
# - Where to Log -
#log_destination = 'stderr'		# Valid values are combinations of
					# stderr, csvlog, syslog, and eventlog,
					# depending on platform.  csvlog
					# requires logging_collector to be on.
# This is used when logging to stderr:
#logging_collector = off		# Enable capturing of stderr and csvlog
					# into log files. Required to be on for
					# csvlogs.
					# (change requires restart)
# These are only used if logging_collector is on:
#log_directory = 'pg_log'		# directory where log files are written,
					# can be absolute or relative to PGDATA
#log_filename = 'postgresql-%Y-%m-%d_%H%M%S.log' # log file name pattern,
					# can include strftime() escapes
#log_file_mode = 0600			# creation mode for log files,
					# begin with 0 to use octal notation
#log_truncate_on_rotation = off		# If on, an existing log file with the
					# same name as the new log file will be
					# truncated rather than appended to.
					# But such truncation only occurs on
					# time-driven rotation, not on restarts
					# or size-driven rotation.  Default is
					# off, meaning append to existing files
					# in all cases.
#log_rotation_age = 1d			# Automatic rotation of logfiles will
					# happen after that time.  0 disables.
#log_rotation_size = 10MB		# Automatic rotation of logfiles will
					# happen after that much log output.
					# 0 disables.
# These are relevant when logging to syslog:
#syslog_facility = 'LOCAL0'
```

```
#syslog_ident = 'postgres'
# This is only relevant when logging to eventlog (win32):
#event_source = 'PostgreSQL'
# - When to Log -
#client_min_messages = notice          # values in order of decreasing detail:
                                      #   debug5
                                      #   debug4
                                      #   debug3
                                      #   debug2
                                      #   debug1
                                      #   log
                                      #   notice
                                      #   warning
                                      #   error
#log_min_messages = warning            # values in order of decreasing detail:
                                      #   debug5
                                      #   debug4
                                      #   debug3
                                      #   debug2
                                      #   debug1
                                      #   info
                                      #   notice
                                      #   warning
                                      #   error
                                      #   log
                                      #   fatal
                                      #   panic
#log_min_error_statement = error      # values in order of decreasing detail:
                                      #   debug5
                                      #   debug4
                                      #   debug3
                                      #   debug2
                                      #   debug1
```

```
                                        #   info
                                        #   notice
                                        #   warning
                                        #   error
                                        #   log
                                        #   fatal
                                        #   panic (effectively off)
#log_min_duration_statement = -1        # -1 is disabled, 0 logs all statements
                                        # and their durations, > 0 logs only
                                        # statements running at least this number
                                        # of milliseconds

# - What to Log -

#debug_print_parse = off
#debug_print_rewritten = off
#debug_print_plan = off
#debug_pretty_print = on
#log_checkpoints = off
#log_connections = off
#log_disconnections = off
#log_duration = off
#log_error_verbosity = default          # terse, default, or verbose messages
#log_hostname = off
log_line_prefix = '%t [%p-%l] %q%u@%d '                 # special values:
                                        #   %a = application name
                                        #   %u = user name
                                        #   %d = database name
                                        #   %r = remote host and port
                                        #   %h = remote host
                                        #   %p = process ID
                                        #   %t = timestamp without milliseconds
                                        #   %m = timestamp with milliseconds
                                        #   %i = command tag
                                        #   %e = SQL state
```

```
                                        #   %c = session ID
                                        #   %l = session line number
                                        #   %s = session start timestamp
                                        #   %v = virtual transaction ID
                                        #   %x = transaction ID (0 if none)
                                        #   %q = stop here in non-session
                                        #        processes
                                        #   %% = '%'
                                        # e.g. '<%u%%%d> '
#log_lock_waits = off                   # log lock waits >= deadlock_timeout
#log_statement = 'none'                         # none, ddl, mod, all
#log_replication_commands = off
#log_temp_files = -1                    # log temporary files equal or larger
                                        # than the specified size in kilobytes;
                                        # -1 disables, 0 logs all temp files
log_timezone = 'localtime'
# - Process Title -
#cluster_name = ''                      # added to process titles if nonempty
                                        # (change requires restart)
#update_process_title = on
#------------------------------------------------------------------------------
# RUNTIME STATISTICS
#------------------------------------------------------------------------------
# - Query/Index Statistics Collector -
#track_activities = on
#track_counts = on
#track_io_timing = off
#track_functions = none                         # none, pl, all
#track_activity_query_size = 1024   # (change requires restart)
stats_temp_directory = '/var/run/postgresql/9.5-main.pg_stat_tmp'
# - Statistics Monitoring -
#log_parser_stats = off
#log_planner_stats = off
```

```
#log_executor_stats = off
#log_statement_stats = off
#------------------------------------------------------------------------------
# AUTOVACUUM PARAMETERS
#------------------------------------------------------------------------------
#autovacuum = on                        # Enable autovacuum subprocess?  'on'
                                        # requires track_counts to also be on.
#log_autovacuum_min_duration = -1       # -1 disables, 0 logs all actions and
                                        # their durations, > 0 logs only
                                        # actions running at least this number
                                        # of milliseconds.
#autovacuum_max_workers = 3             # max number of autovacuum subprocesses
                                        # (change requires restart)
#autovacuum_naptime = 1min              # time between autovacuum runs
#autovacuum_vacuum_threshold = 50       # min number of row updates before
                                        # vacuum
#autovacuum_analyze_threshold = 50      # min number of row updates before
                                        # analyze
#autovacuum_vacuum_scale_factor = 0.2   # fraction of table size before vacuum
#autovacuum_analyze_scale_factor = 0.1  # fraction of table size before analyze
#autovacuum_freeze_max_age = 200000000  # maximum XID age before forced
vacuum
                                        # (change requires restart)
#autovacuum_multixact_freeze_max_age = 400000000    # maximum multixact age
                                        # before forced vacuum
                                        # (change requires restart)
#autovacuum_vacuum_cost_delay = 20ms    # default vacuum cost delay for
                                        # autovacuum, in milliseconds;
                                        # -1 means use vacuum_cost_delay
#autovacuum_vacuum_cost_limit = -1      # default vacuum cost limit for
                                        # autovacuum, -1 means use
                                        # vacuum_cost_limit
#------------------------------------------------------------------------------
```

```
# CLIENT CONNECTION DEFAULTS
#------------------------------------------------------------------------------
# - Statement Behavior -
#search_path = '"$user", public'        # schema names
#default_tablespace = ''                # a tablespace name, '' uses the default
#temp_tablespaces = ''                  # a list of tablespace names, '' uses
                                        # only default tablespace
#check_function_bodies = on
#default_transaction_isolation = 'read committed'
#default_transaction_read_only = off
#default_transaction_deferrable = off
#session_replication_role = 'origin'
#statement_timeout = 0                  # in milliseconds, 0 is disabled
#lock_timeout = 0                       # in milliseconds, 0 is disabled
#vacuum_freeze_min_age = 50000000
#vacuum_freeze_table_age = 150000000
#vacuum_multixact_freeze_min_age = 5000000
#vacuum_multixact_freeze_table_age = 150000000
#bytea_output = 'hex'                   # hex, escape
#xmlbinary = 'base64'
#xmloption = 'content'
#gin_fuzzy_search_limit = 0
#gin_pending_list_limit = 4MB
# - Locale and Formatting -
datestyle = 'iso, mdy'
#intervalstyle = 'postgres'
timezone = 'localtime'
#timezone_abbreviations = 'Default'     # Select the set of available time zone
                                        # abbreviations.  Currently, there are
                                        #   Default
                                        #   Australia (historical usage)
                                        #   India
                                        # You can create your own file in
```

```
                                        # share/timezonesets/.
#extra_float_digits = 0                 # min -15, max 3
#client_encoding = sql_ascii            # actually, defaults to database
                                        # encoding
# These settings are initialized by initdb, but they can be changed.
lc_messages = 'en_US.UTF-8'                     # locale for system error message
                                        # strings
lc_monetary = 'en_US.UTF-8'                     # locale for monetary formatting
lc_numeric = 'en_US.UTF-8'              # locale for number formatting
lc_time = 'en_US.UTF-8'                 # locale for time formatting
# default configuration for text search
default_text_search_config = 'pg_catalog.english'
# - Other Defaults -
#dynamic_library_path = '$libdir'
#local_preload_libraries = ''
#session_preload_libraries = ''
#------------------------------------------------------------------------------
# LOCK MANAGEMENT
#------------------------------------------------------------------------------
#deadlock_timeout = 1s
#max_locks_per_transaction = 64          # min 10
                                        # (change requires restart)
#max_pred_locks_per_transaction = 64     # min 10
                                        # (change requires restart)
#------------------------------------------------------------------------------
# VERSION/PLATFORM COMPATIBILITY
#------------------------------------------------------------------------------
# - Previous PostgreSQL Versions -
#array_nulls = on
#backslash_quote = safe_encoding   # on, off, or safe_encoding
#default_with_oids = off
#escape_string_warning = on
#lo_compat_privileges = off
```

```
#operator_precedence_warning = off
#quote_all_identifiers = off
#sql_inheritance = on
#standard_conforming_strings = on
#synchronize_seqscans = on
# - Other Platforms and Clients -
#transform_null_equals = off
#------------------------------------------------------------------------------
# ERROR HANDLING
#------------------------------------------------------------------------------
#exit_on_error = off                    # terminate session on any error?
#restart_after_crash = on               # reinitialize after backend crash?
#------------------------------------------------------------------------------
# CONFIG FILE INCLUDES
#------------------------------------------------------------------------------
# These options allow settings to be loaded from files other than the
# default postgresql.conf.
#include_dir = 'conf.d'                         # include files ending in '.conf' from
                                                # directory 'conf.d'
#include_if_exists = 'exists.conf'      # include file only if it exists
#include = 'special.conf'               # include file
#------------------------------------------------------------------------------
# CUSTOMIZED OPTIONS
#------------------------------------------------------------------------------
# Add settings for extensions here
```

Příloha 2 - Konfigurační soubor databázového systému MySQL

```
# The MySQL database server configuration file.
# You can copy this to one of:
# - "/etc/mysql/my.cnf" to set global options,
# - "~/.my.cnf" to set user-specific options.
# One can use all long options that the program supports.
```

```
# Run program with --help to get a list of available options and with
# --print-defaults to see which it would actually understand and use.
# For explanations see
# http://dev.mysql.com/doc/mysql/en/server-system-variables.html
# This will be passed to all mysql clients
# It has been reported that passwords should be enclosed with ticks/quotes
# escpecially if they contain "#" chars...
# Remember to edit /etc/mysql/debian.cnf when changing the socket location.
# Here is entries for some specific programs
# The following values assume you have at least 32M ram
[mysqld_safe]
socket          = /var/run/mysqld/mysqld.sock
nice            = 0
[mysqld]
# * Basic Settings
user            = mysql
pid-file        = /var/run/mysqld/mysqld.pid
socket          = /var/run/mysqld/mysqld.sock
port            = 3306
basedir         = /usr
datadir         = /var/lib/mysql
tmpdir          = /tmp
lc-messages-dir         = /usr/share/mysql
skip-external-locking
# Instead of skip-networking the default is now to listen only on
# localhost which is more compatible and is not less secure.
bind-address            = 127.0.0.1
# * Fine Tuning
key_buffer_size                 = 16M
max_allowed_packet = 16M
thread_stack            = 192K
thread_cache_size       = 8
# This replaces the startup script and checks MyISAM tables if needed
```

```
# the first time they are touched
myisam-recover-options  = BACKUP
#max_connections        = 100
#table_cache            = 64
#thread_concurrency     = 10
# * Query Cache Configuration
query_cache_limit      = 1M
query_cache_size        = 16M
# * Logging and Replication
# Both location gets rotated by the cronjob.
# Be aware that this log type is a performance killer.
# As of 5.1 you can enable the log at runtime!
#general_log_file        = /var/log/mysql/mysql.log
#general_log             = 1
# Error log - should be very few entries.
log_error = /var/log/mysql/error.log
# Here you can see queries with especially long duration
#log_slow_queries      = /var/log/mysql/mysql-slow.log
#long_query_time = 2
#log-queries-not-using-indexes
# The following can be used as easy to replay backup logs or for replication.
# note: if you are setting up a replication slave, see README.Debian about
#       other settings you may need to change.
#server-id               = 1
#log_bin                        = /var/log/mysql/mysql-bin.log
expire_logs_days       = 10
max_binlog_size   = 100M
#binlog_do_db                   = include_database_name
#binlog_ignore_db     = include_database_name
# * InnoDB
# InnoDB is enabled by default with a 10MB datafile in /var/lib/mysql/.
# Read the manual for more InnoDB related options. There are many!
# * Security Features
```

```
# Read the manual, too, if you want chroot!
# chroot = /var/lib/mysql/
# For generating SSL certificates I recommend the OpenSSL GUI "tinyca".
# ssl-ca=/etc/mysql/cacert.pem
# ssl-cert=/etc/mysql/server-cert.pem
# ssl-key=/etc/mysql/server-key.pem
```

Příloha 3 - Databázové dotazy – PostgreSQL

1. Najdi jedinečné hodnoty typu budov.

```
SELECT type FROM osm_new_buildings GROUP BY type;
```

2. Najdi id a název vodních toků vzestupně setříděné podle jména.

```
SELECT id, name, type FROM osm_new_waterways ORDER BY name;
```

3. Najdi id a název vodních segmentů, které mají název „Elbe“.

```
SELECT id, name FROM osm_new_waterways WHERE name='Elbe';
```

4. Smaž všechny popisky, které mají typ „village“.

```
DELETE FROM osm_new_places WHERE type='village';
```

5. Vlož záznamy z hlavních cest a motorových cest do nové tabulky.

Tvorba nové tabulky s názvem „comunication“

```
CREATE TABLE comunication (id integer NOT NULL,osm_id
bigint,name character varying(255),type character
varying(255),tunnel smallint,bridge smallint,oneway
smallint,z_order smallint,length real,geometry
geometry(LineString,900913));
ALTER TABLE comunication OWNER TO root;
CREATE SEQUENCE comunication_id_seq START WITH 1 INCREMENT BY
1 NO MINVALUE NO MAXVALUE CACHE 1;
ALTER TABLE comunication_id_seq OWNER TO root;
ALTER SEQUENCE comunication_id_seq OWNED BY comunication.id;
ALTER TABLE ONLY comunication ALTER COLUMN id SET DEFAULT
nextval('comunication_id_seq'::regclass);
ALTER TABLE ONLY comunication ADD CONSTRAINT
comunication_pkey PRIMARY KEY (id);
```

```
CREATE INDEX comunication_geom ON comunication USING gist
(geometry);
```

Naplnění tabulky

```
INSERT INTO comunication (geometry, osm_id, name, type,
tunnel, bridge, oneway, z_order) SELECT ST_AsText(geometry),
osm_id, name, type, tunnel, bridge, oneway, z_order FROM
osm_new_mainroads UNION SELECT ST_AsText(geometry), osm_id,
name, type, tunnel, bridge, oneway, z_order FROM
osm_new_motorways;
```

6. Najdi všechny id tabulky comunication a id tabulky hlavních cest, které jsou společně přirozeně spojené s tabulkou comunication přes atributy osm_id.

```
SELECT c.id, mw.id FROM comunication c INNER JOIN
osm_new_motorways mw ON c.osm_id = mw.osm_id;
```

7. Najdi všechny id a názvy škol z tabulky veřejných zařízení v MBR 52,5 s.š. 13,35 v.d. a 52,55 s.š. 13,4 v.d.

```
SELECT id, name FROM osm_new_amenities WHERE type='school'
AND (geometry &&
ST_MakeEnvelope(1486115.20209,6891041.72389,1491681.17663,690
0190.04114,900913));
```

8. Najdi všechny id a názvy hotelů z tabulky budov do vzdálenosti 10 000m z bodu 52,5 s.š. a 13,4 v.d.

```
SELECT id, name, type FROM osm_new_buildings WHERE
type='hotel' AND ST_DWithin(geometry,
ST_GeomFromText('POINT(1491681.17663 6891041.72389)',
900913), 10000);
```

9. Vypočti rozlohu vodních ploch.

```
SELECT ST_Area(geometry) FROM osm_new_waterareas;
```

10. Zapiš do nového sloupce délky všech segmentů silnic.

```
UPDATE comunication SET length=ST_Length(geometry);
```

11. Vytvoř buffer leteckých cest velikosti 50 m.

```
SELECT ST_Buffer(geometry, 50.0) FROM osm_new_aeroways;
```

12. Vypočti centroid transportních oblastí.

```
SELECT ST_Centroid(geometry) FROM osm_new_transport_areas;
```

13. Vyber geometrii železnic transformovanou do formátu WKT.

```
SELECT ST_AsText(geometry) FROM osm_new_railways;
```

14. Najdi id, názvy a typy veřejných zařízení , které neleží ve spolkové zemi „Hamburg“.

```
SELECT osm.id, osm.name, osm.type FROM osm_new_amenities osm,
    osm_new_admin adm WHERE adm.name = 'Hamburg' AND
        ST_Disjoint(osm.geometry, adm.geometry);
```

15. Najdi id všech přítoků řeky „Elbe“.

```
SELECT osm.id FROM osm_new_waterways osm, osm_new_waterways
    ww WHERE ww.name='Elbe' AND ST_Touches(osm.geometry,
                ww.geometry);
```

16. Najdi id a název vodních ploch, které alespoň částí překrývají přírodní rezervace.

```
SELECT wa.id, wa.name FROM osm_new_landusages osm,
  osm_new_waterareas wa WHERE osm.type='nature_reserve' AND
        ST_Overlaps(osm.geometry, wa.geometry);
```

17. Najdi id a název vodních ploch, které celé leží v krajinném pokryvu typu „forest“.

```
SELECT osm.id, osm.name, osm.type FROM osm_new_waterareas
  osm, osm_new_landusages lu WHERE lu.type = 'forest' AND
        ST_Contains(osm.geometry, lu.geometry);
```

18. Najdi id a název přírodních rezervací, které se shodují s krajinným pokryvem typu „grass“.

```
SELECT osm.id, osm.name FROM osm_new_landusages osm,
    osm_new_landusages lu WHERE ST_Equals(osm.geometry,
        lu.geometry) AND lu.type='grass' AND
            osm.type='nature_reserve';
```

19. Najdi všechny oblasti, na kterých se překrývají plochy krajinného pokryvu typů „nature_reserve“ a „forest“.

```
SELECT ST_Intersection(osm.geometry, lu.geometry) FROM
    osm_new_landusages osm, osm_new_landusages lu WHERE
        ST_Intersects(osm.geometry,lu.geometry) AND
      osm.type='nature_reserve' AND lu.type='forest';
```

20. Spoj geometrii bodů veřejných zařízení typu „university“ a „school“.

```
SELECT ST_Union(osm.geometry, r.geometry) FROM
```

```
osm_new_amenities osm, osm_new_amenities r WHERE
osm.type='university' AND r.type='school';
```

21. Najdi části hospodářských oblastí, které nezasahují do oblastí tráv, ale současně se s něma protínají. (difference)

```
SELECT ST_Difference(osm.geometry, lu.geometry) FROM
osm_new_landusages osm, osm_new_landusages lu WHERE
osm.type='farmland' AND lu.type='grass' AND
ST_Intersects(osm.geometry, lu.geometry);
```

22. Vypočti hustotu dálnic v jednotlivých okresech.

```
SELECT
(SUM(ST_Length(ST_Intersection(mw.geometry,a.geometry)))/SUM(
ST_Area(a.geometry))) as hustota,a.name FROM
osm_new_motorways mw, osm_new_admin a WHERE a.admin_level='6'
AND ST_Intersects(mw.geometry, a.geometry) GROUP BY a.name;
```

23. Vypočti hustotu dálnic v přepočtu na obyvatele.

```
SELECT (SUM(ST_Length(geometry))/80620000) FROM
osm_new_motorways;
```

24. Vypočti rozlohu jednotlivých krajinných pokryvů v okrese „Worms“.

```
SELECT SUM(ST_Area(ST_Intersection(lu.geometry, a.geometry)))
FROM osm_new_admin a, osm_new_landusages lu WHERE
a.name='Worms' AND ST_Intersects(lu.geometry, a.geometry)
GROUP BY lu.type;
```

25. Vypočti míru ekologické stability krajiny v okrese „Worms“.

```
SELECT ((SUM(ST_Area(ST_Intersection(gr.geometry,
a.geometry))) + SUM(ST_Area(ST_Intersection(f.geometry,
a.geometry))) + SUM(ST_Area(ST_Intersection(wa.geometry,
a.geometry)))) / (SUM(ST_Area(ST_Intersection(res.geometry,
a.geometry))) + SUM(ST_Area(ST_Intersection()) FROM
osm_new_admin a, osm_new_landusages gr, osm_new_landusages f,
osm_new_landusages res, osm_new_waterareas WHERE
a.name='Worms' AND ST_Intersects(f.geometry, a.geometry) AND
ST_Intersects(wa.geometry, a.geometry) AND
ST_Intersects(res.geometry, a.geometry) AND (f.type='forest'
```

```
OR f.type='grass' OR f.type='park' OR f.type='scrub' OR
f.type='wood') AND (res.type='residence' OR
res.type='commercial' OR res.type='industrial' OR
res.type='parking' OR res.type='farmland');
```

Příloha 4 - Databázové dotazy – PostgreSQL

1. Najdi jedinečné hodnoty typu budov.

```
SELECT type FROM osm_new_buildings GROUP BY type;
```

2. Najdi id a název vodních toků vzestupně setříděné podle jména.

```
SELECT id, name, type FROM osm_new_waterways ORDER BY name;
```

3. Najdi id a název vodních segmentů, které mají název „Elbe“.

```
SELECT id, name FROM osm_new_waterways WHERE name='Elbe';
```

4. Smaž všechny popisky, které mají typ „village“.

```
DELETE FROM osm_new_places WHERE type='village';
```

5. Vlož záznamy z hlavních cest a motorových cest do nové tabulky.

Tvorba nové tabulky s názvem „comunication“

```
CREATE TABLE 'comunication' (id INTEGER PRIMARY KEY,'osm_id'
BIGINT,'name' VARCHAR(255),'type' VARCHAR(255),'tunnel'
INTEGER_INT16,'bridge' INTEGER_INT16,'oneway'
INTEGER_INT16,'length' FLOAT,'z_order' INTEGER_INT16);
SELECT AddGeometryColumn('comunication', 'geometry', 3785,
'LINESTRING',2) FROM comunication;
CREATE TRIGGER "ggi_comunication_GEOMETRY" BEFORE INSERT ON
"comunication" FOR EACH ROW BEGIN SELECT RAISE(ROLLBACK,
'comunication.GEOMETRY violates Geometry constraint [geom-
type or SRID not allowed]') WHERE (SELECT geometry_type FROM
geometry_columns WHERE Lower(f_table_name) =
Lower('comunication') AND Lower(f_geometry_column) =
Lower('GEOMETRY') AND GeometryConstraints(NEW."GEOMETRY",
geometry_type, srid) = 1) IS NULL; END;
CREATE TRIGGER "ggu_comunication_GEOMETRY" BEFORE UPDATE
OF "GEOMETRY" ON "comunication" FOR EACH ROW BEGIN SELECT
RAISE(ROLLBACK, 'comunication.GEOMETRY violates Geometry
constraint [geom-type or SRID not allowed]') WHERE (SELECT
```

```
geometry_type FROM geometry_columns WHERE Lower(f_table_name)
    = Lower('comunication') AND Lower(f_geometry_column) =
  Lower('GEOMETRY') AND GeometryConstraints(NEW."GEOMETRY",
          geometry_type, srid) = 1) IS NULL; END;
     CREATE TRIGGER "gid_comunication_GEOMETRY" AFTER DELETE
      ON "comunication" FOR EACH ROW BEGIN DELETE FROM
   "idx_comunication_GEOMETRY" WHERE pkid=OLD.ROWID; END;
     CREATE TRIGGER "gii_comunication_GEOMETRY" AFTER INSERT
      ON "comunication" FOR EACH ROW BEGIN DELETE FROM
   "idx_comunication_GEOMETRY" WHERE pkid=NEW.ROWID; SELECT
     RTreeAlign('idx_comunication_GEOMETRY', NEW.ROWID,
                  NEW."GEOMETRY"); END;
     CREATE TRIGGER "giu_comunication_GEOMETRY" AFTER UPDATE
  OF "GEOMETRY" ON "comunication" FOR EACH ROW BEGIN DELETE
FROM "idx_comunication_GEOMETRY" WHERE pkid=NEW.ROWID; SELECT
     RTreeAlign('idx_comunication_GEOMETRY', NEW.ROWID,
                  NEW."GEOMETRY"); END;
     CREATE TRIGGER "tmd_comunication_GEOMETRY" AFTER DELETE
         ON "comunication" FOR EACH ROW BEGIN UPDATE
           geometry_columns_time SET last_delete =
         strftime('%Y-%m-%dT%H:%M:%fZ', 'now') WHERE
       Lower(f_table_name) = Lower('comunication') AND
     Lower(f_geometry_column) = Lower('GEOMETRY'); END;
     CREATE TRIGGER "tmi_comunication_GEOMETRY" AFTER INSERT
         ON "comunication" FOR EACH ROW BEGIN UPDATE
           geometry_columns_time SET last_insert =
         strftime('%Y-%m-%dT%H:%M:%fZ', 'now') WHERE
       Lower(f_table_name) = Lower('comunication') AND
     Lower(f_geometry_column) = Lower('GEOMETRY'); END;
     CREATE TRIGGER "tmu_comunication_GEOMETRY" AFTER UPDATE
         ON "comunication" FOR EACH ROW BEGIN UPDATE
           geometry_columns_time SET last_update =
         strftime('%Y-%m-%dT%H:%M:%fZ', 'now') WHERE
```

```
Lower(f_table_name) = Lower('comunication') AND
Lower(f_geometry_column) = Lower('GEOMETRY'); END;
```

Naplnění tabulky

```
INSERT INTO comunication (geometry, osm_id, name, type,
tunnel, bridge, oneway, z_order) SELECT geometry, osm_id,
name, type, tunnel, bridge, oneway, z_order FROM
osm_new_mainroads UNION SELECT geometry, osm_id, name, type,
tunnel, bridge, oneway, z_order FROM osm_new_motorways;
```

6. Najdi všechny id tabulky comunication a id tabulky hlavních cest, které jsou společně přirozeně spojené s tabulkou comunication přes atributy osm_id.

```
SELECT c.id, mw.id FROM comunication c INNER JOIN
osm_new_motorways mw ON c.osm_id = mw.osm_id;
```

7. Najdi všechny id a názvy škol z tabulky veřejných zařízení v MBR 52,5 s.š. 13,35 v.d. a 52,55 s.š. 13,4 v.d.

```
SELECT id, name FROM osm_new_amenities WHERE type='school'
AND MbrWithin(geometry,
ST_GeomFromText('POLYGON((1486115.20209 6891041.72389,
1486115.20209 6900190.04114, 1491681.17663 6900190.04114,
1491681.17663 6891041.72389, 1486115.20209 6891041.72389))',
900913));
```

8. Najdi všechny id a názvy hotelů z tabulky budov do vzdálenosti 10 000m z bodu 52,5 s.š. a 13,4 v.d.

```
SELECT id, name, type FROM osm_new_buildings WHERE
type='hotel' AND PtDistWithin(geometry,
ST_GeomFromText('POINT(1491681.17663 6891041.72389)',
900913), 10000.0);
```

9. Vypočti rozlohu vodních ploch.

```
SELECT ST_Area(geometry) FROM osm_new_waterareas;
```

10. Zapiš do nového sloupce délky všech segmentů silnic.

```
UPDATE comunication SET length=ST_Length(geometry);
```

11. Vytvoř buffer leteckých cest velikosti 50 m.

```
SELECT ST_Buffer(geometry, 50.0) FROM osm_new_aeroways;
```

12. Vypočti centroid transportních oblastí.

```
SELECT ST_Centroid(geometry) FROM osm_new_transport_areas;
```

13. Vyber geometrii železnic transformovanou do formátu WKT.

```
SELECT ST_AsText(geometry) FROM osm_new_railways;
```

14. Najdi id, názvy a typy veřejných zařízení , které neleží ve spolkové zemi „Hamburg“.

```
SELECT osm.id, osm.name, osm.type FROM osm_new_amenities osm,
osm_new_admin adm WHERE adm.name = 'Hamburg' AND
ST_Disjoint(osm.geometry, adm.geometry);
```

15. Najdi id všech přítoků řeky „Elbe“.

```
SELECT osm.id FROM osm_new_waterways osm, osm_new_waterways
ww WHERE ww.name='Elbe' AND ST_Touches(osm.geometry,
ww.geometry);
```

16. Najdi id a název vodních ploch, které alespoň částí překrývají přírodní rezervace.

```
SELECT wa.id, wa.name FROM osm_new_landusages osm,
osm_new_waterareas wa WHERE osm.type='nature_reserve' AND
ST_Overlaps(osm.geometry, wa.geometry);
```

17. Najdi id a název vodních ploch, které celé leží v krajinném pokryvu typu „forest“.

```
SELECT osm.id, osm.name, osm.type FROM osm_new_waterareas
osm, osm_new_landusages lu WHERE lu.type = 'forest' AND
ST_Contains(osm.geometry, lu.geometry);
```

18. Najdi id a název přírodních rezervací, které se shodují s krajinným pokryvem typu „grass“.

```
SELECT osm.id, osm.name FROM osm_new_landusages osm,
osm_new_landusages lu WHERE Equals(osm.geometry, lu.geometry)
AND lu.type='grass' AND osm.type='nature_reserve';
```

19. Najdi všechny oblasti, na kterých se překrývají plochy krajinného pokryvu typů „nature_reserve“ a „forest“.

```
SELECT ST_Intersection(osm.geometry, lu.geometry) FROM
osm_new_landusages osm, osm_new_landusages lu WHERE
ST_Intersects(osm.geometry,lu.geometry) AND
osm.type='nature_reserve' AND lu.type='forest';
```

20. Spoj geometrii bodů veřejných zařízení typu „university“ a „school“.

```
SELECT ST_Union(osm.geometry, r.geometry) FROM
```

```
osm_new_amenities osm, osm_new_amenities r WHERE
osm.type='university' AND r.type='school';
```

21. Najdi části hospodářských oblastí, které nezasahují do oblastí tráv, ale současně se s něma protínají. (difference)

```
SELECT ST_Difference(osm.geometry, lu.geometry) FROM
osm_new_landusages osm, osm_new_landusages lu WHERE
osm.type='farmland' AND lu.type='grass' AND
ST_Intersects(osm.geometry, lu.geometry);
```

22. Vypočti hustotu dálnic v jednotlivých okresech.

```
SELECT
(SUM(ST_Length(ST_Intersection(mw.geometry,a.geometry)))/SUM(
ST_Area(a.geometry))) as hustota,a.name FROM
osm_new_motorways mw, osm_new_admin a WHERE a.admin_level='6'
AND ST_Intersects(mw.geometry, a.geometry) GROUP BY a.name;
```

23. Vypočti hustotu dálnic v přepočtu na obyvatele.

```
SELECT (SUM(ST_Length(geometry))/80620000) FROM
osm_new_motorways;
```

24. Vypočti rozlohu jednotlivých krajinných pokryvů v okrese „Worms“.

```
SELECT SUM(ST_Area(ST_Intersection(lu.geometry, a.geometry)))
FROM osm_new_admin a, osm_new_landusages lu WHERE
a.name='Worms' AND ST_Intersects(lu.geometry, a.geometry)
GROUP BY lu.type;
```

25. Vypočti míru ekologické stability krajiny v okrese „Worms“.

```
SELECT ((SUM(ST_Area(ST_Intersection(gr.geometry,
a.geometry))) + SUM(ST_Area(ST_Intersection(f.geometry,
a.geometry))) + SUM(ST_Area(ST_Intersection(wa.geometry,
a.geometry)))) / (SUM(ST_Area(ST_Intersection(res.geometry,
a.geometry))) + SUM(ST_Area(ST_Intersection()) FROM
osm_new_admin a, osm_new_landusages gr, osm_new_landusages f,
osm_new_landusages res, osm_new_waterareas WHERE
a.name='Worms' AND ST_Intersects(f.geometry, a.geometry) AND
ST_Intersects(wa.geometry, a.geometry) AND
ST_Intersects(res.geometry, a.geometry) AND (f.type='forest'
```

```
OR f.type='grass' OR f.type='park' OR f.type='scrub' OR
f.type='wood') AND (res.type='residence' OR
res.type='commercial' OR res.type='industrial' OR
res.type='parking' OR res.type='farmland');
```

Příloha 5 - Databázové dotazy – PostgreSQL

1. Najdi jedinečné hodnoty typu budov.

```
SELECT type FROM osm_new_buildings GROUP BY type;
```

2. Najdi id a název vodních toků vzestupně setříděné podle jména.

```
SELECT id, name, type FROM osm_new_waterways ORDER BY name;
```

3. Najdi id a název vodních segmentů, které mají název „Elbe“.

```
SELECT id, name FROM osm_new_waterways WHERE name='Elbe';
```

4. Smaž všechny popisky, které mají typ „village“.

```
DELETE FROM osm_new_places WHERE type='village';
```

5. Vlož záznamy z hlavních cest a motorových cest do nové tabulky.

Tvorba nové tabulky s názvem „comunication“

```
CREATE TABLE `comunication` (`id` int(11) NOT NULL
AUTO_INCREMENT, `SHAPE` geometry NOT NULL, `osm_id`
bigint(20) DEFAULT NULL, `name` varchar(255) DEFAULT NULL,
`type` varchar(255) DEFAULT NULL, `tunnel` decimal(5,0)
DEFAULT NULL, `bridge` decimal(5,0) DEFAULT NULL, `oneway`
decimal(5,0) DEFAULT NULL, `length` FLOAT, `z_order`
decimal(5,0) DEFAULT NULL, UNIQUE KEY `id` (`id`), SPATIAL
KEY `SHAPE` (`SHAPE`)) ENGINE=InnoDB AUTO_INCREMENT=10588
DEFAULT CHARSET=utf8;
```

Naplnění tabulky

```
INSERT INTO comunication (ST_AsText(SHAPE), osm_id, name,
type, tunnel, bridge, oneway, z_order) SELECT SHAPE, osm_id,
name, type, tunnel, bridge, oneway, z_order FROM
osm_new_mainroads UNION SELECT ST_AsText(SHAPE), osm_id,
name, type, tunnel, bridge, oneway, z_order FROM
osm_new_motorways;
```

6. Najdi všechny id tabulky comunication a id tabulky hlavních cest, které jsou

společně přirozeně spojené s tabulkou comunication přes atributy osm_id.

```
SELECT c.id, mw.id FROM comunication c INNER JOIN
osm_new_motorways mw ON c.osm_id = mw.osm_id;
```

7. Najdi všechny id a názvy škol z tabulky veřejných zařízení v MBR 52,5 s.š. 13,35 v.d. a 52,55 s.š. 13,4 v.d.

```
SELECT id, name FROM osm_new_amenities WHERE type='school'
AND MbrWithin(SHAPE, ST_GeomFromText('POLYGON((1486115.20209
6891041.72389, 1486115.20209 6900190.04114, 1491681.17663
6900190.04114, 1491681.17663 6891041.72389, 1486115.20209
6891041.72389))', 900913));
```

8. Najdi všechny id a názvy hotelů z tabulky budov do vzdálenosti 10 000m z bodu 52,5 s.š. a 13,4 v.d.

```
SELECT id, name, type FROM osm_new_buildings WHERE
type='hotel' AND ST_Within(SHAPE,
ST_Buffer(ST_GeomFromText('POINT(1491681.17663
6891041.72389)', 900913), 10000.0));
```

9. Vypočti rozlohu vodních ploch.

```
SELECT ST_Area(SHAPE) FROM osm_new_waterareas;
```

10. Zapiš do nového sloupce délky všech segmentů silnic.

```
UPDATE comunication SET length=ST_Length(SHAPE);
```

11. Vytvoř buffer leteckých cest velikosti 50 m.

```
SELECT ST_Buffer(SHAPE, 50.0) FROM osm_new_aeroways;
```

12. Vypočti centroid transportních oblastí.

```
SELECT ST_Centroid(SHAPE) FROM osm_new_transport_areas;
```

13. Vyber geometrii železnic transformovanou do formátu WKT.

```
SELECT ST_AsText(SHAPE) FROM osm_new_railways;
```

14. Najdi id, názvy a typy veřejných zařízení , které neleží ve spolkové zemi „Hamburg“.

```
SELECT osm.id, osm.name, osm.type FROM osm_new_amenities osm,
osm_new_admin adm WHERE adm.name = 'Hamburg' AND
ST_Disjoint(osm.SHAPE, adm.SHAPE);
```

15. Najdi id všech přítoků řeky „Elbe“.

```
SELECT osm.id FROM osm_new_waterways osm, osm_new_waterways
```

```
ww WHERE ww.name='Elbe' AND ST_Touches(osm.SHAPE, ww.SHAPE);
```

16. Najdi id a název vodních ploch, které alespoň částí překrývají přírodní rezervace.

```
SELECT wa.id, wa.name FROM osm_new_landusages osm,
osm_new_waterareas wa WHERE osm.type='nature_reserve' AND
ST_Overlaps(osm.SHAPE, wa.SHAPE);
```

17. Najdi id a název vodních ploch, které celé leží v krajinném pokryvu typu „forest“.

```
SELECT osm.id, osm.name, osm.type FROM osm_new_waterareas
osm, osm_new_landusages lu WHERE lu.type = 'forest' AND
ST_Contains(osm.SHAPE, lu.SHAPE);
```

18. Najdi id a název přírodních rezervací, které se shodují s krajinným pokryvem typu „grass“.

```
SELECT osm.id, osm.name FROM osm_new_landusages osm,
osm_new_landusages lu WHERE ST_Equals(osm.SHAPE, lu.SHAPE)
AND lu.type='grass' AND osm.type='nature_reserve';
```

19. Najdi všechny oblasti, na kterých se překrývají plochy krajinného pokryvu typů „nature_reserve“ a „forest“.

```
SELECT ST_Intersection(osm.SHAPE, lu.SHAPE) FROM
osm_new_landusages osm, osm_new_landusages lu WHERE
ST_Intersects(osm.SHAPE,lu.SHAPE) AND
osm.type='nature_reserve' AND lu.type='forest';
```

20. Spoj geometrii bodů veřejných zařízení typu „university“ a „school“.

```
SELECT ST_Union(osm.SHAPE, r.SHAPE) FROM osm_new_amenities
osm, osm_new_amenities r WHERE osm.type='university' AND
r.type='school';
```

21. Najdi části hospodářských oblastí, které nezasahují do oblastí tráv, ale současně se s něma protínají. (difference)

```
SELECT ST_Difference(osm.SHAPE, lu.SHAPE) FROM
osm_new_landusages osm, osm_new_landusages lu WHERE
osm.type='farmland' AND lu.type='grass' AND
ST_Intersects(osm.SHAPE, lu.SHAPE);
```

22. Vypočti hustotu dálnic v jednotlivých okresech.

```
SELECT
(SUM(ST_Length(ST_Intersection(mw.SHAPE,a.SHAPE)))/SUM(ST_Are
```

```
a(a.SHAPE))) as hustota,a.name FROM osm_new_motorways mw,
        osm_new_admin a WHERE a.admin_level='6' AND
      ST_Intersects(mw.SHAPE, a.SHAPE) GROUP BY a.name;
```

23. Vypočti hustotu dálnic v přepočtu na obyvatele.

```
SELECT (SUM(ST_Length(SHAPE))/80620000) FROM
                osm_new_motorways;
```

24. Vypočti rozlohu jednotlivých krajinných pokryvů v okrese „Worms“.

```
SELECT SUM(ST_Area(ST_Intersection(lu.SHAPE, a.SHAPE))) FROM
osm_new_admin a, osm_new_landusages lu WHERE a.name='Worms'
  AND ST_Intersects(lu.SHAPE, a.SHAPE) GROUP BY lu.type;
```

25. Vypočti míru ekologické stability krajiny v okrese „Worms“.

```
SELECT ((SUM(ST_Area(ST_Intersection(gr.SHAPE, a.SHAPE))) +
     SUM(ST_Area(ST_Intersection(f.SHAPE, a.SHAPE))) +
     SUM(ST_Area(ST_Intersection(wa.SHAPE, a.SHAPE)))) /
     (SUM(ST_Area(ST_Intersection(res.SHAPE, a.SHAPE))) +
     SUM(ST_Area(ST_Intersection()) FROM osm_new_admin a,
        osm_new_landusages gr, osm_new_landusages f,
       osm_new_landusages res, osm_new_waterareas WHERE
   a.name='Worms' AND ST_Intersects(f.SHAPE, a.SHAPE) AND
ST_Intersects(wa.SHAPE, a.SHAPE) AND ST_Intersects(res.SHAPE,
     a.SHAPE) AND (f.type='forest' OR f.type='grass' OR
     f.type='park' OR f.type='scrub' OR f.type='wood') AND
      (res.type='residence' OR res.type='commercial' OR
       res.type='industrial' OR res.type='parking' OR
                  res.type='farmland');
```